International Research Journal of Innovative Engineering (IRJIE)

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

Developing Large Language Model Integrated Cloud-Native DevSecOps for Software Delivery and Continuous Security Validation

Dr.Vimal Raja Gopinathan

Senior Principal Consultant, Oracle Financial Service Software Ltd, Washington, USA

ABSTRACT: Modern software delivery increasingly relies on cloud-native architectures, containerization, and automated Continuous Integration and Continuous Deployment (CI/CD) pipelines to achieve rapid feature deployment. However, integrating security seamlessly into these high-speed environments remains a critical operational challenge. Traditional DevSecOps practices often depend on static analysis, rule-based scanners, and manual code reviews, which struggle to keep pace with rapid deployment cycles and frequently produce high rates of false positives. This paper presents an integrated framework that embeds Large Language Models (LLMs) directly into cloud-native DevSecOps workflows to enable continuous, real-time security validation and automated remediation. By leveraging fine-tuned LLMs trained on domain-specific security vulnerabilities and cloud infrastructure configurations, the proposed architecture automates threat modeling, static and dynamic application security testing (SAST/DAST) output contextualization, Infrastructure as Code (IaC) compliance verification, and dynamic patch generation. Implemented within a Kubernetes-orchestrated cloud-native ecosystem, the system acts as an intelligent security agent, validating code and infrastructure definitions prior to production deployment. Empirical evaluations demonstrate significant reductions in time-to-remediate (TTR), lowered false-positive overhead, and improved continuous compliance without compromising software delivery speed. Ultimately, this research provides a scalable blueprint for shifting security further left while automating continuous security validation across cloud-native software supply chains.

KEYWORDS: DevSecOps, Large Language Models, Cloud-Native, Continuous Security Validation, Infrastructure as Code, Software Supply Chain Security, Automation.

I. INTRODUCTION

The continuous evolution of cloud computing has fundamentally transformed the modern software development lifecycle, shifting paradigms from monolithic applications to microservices, containerization, and dynamic serverless architectures. To support this rapid velocity, organizations have widely adopted Continuous Integration and Continuous Deployment (CI/CD) pipelines, enabling developer teams to deploy code into production multiple times per day. However, this acceleration in software delivery has created a glaring asymmetry between development speed and security assurance. Traditional security models, which historically relied on perimeter defenses and periodic manual audits, are inherently ill-suited for ephemeral, highly distributed cloud environments. To address this friction, DevSecOps emerged as an operational philosophy aiming to "shift security left" by integrating automated security checks directly into early stage development and deployment workflows. Despite its theoretical promise, standard DevSecOps practices frequently suffer from operational bottlenecks, high noise-to-signal ratios, and an escalating shortage of domain-expert security professionals.

At the core of contemporary DevSecOps friction lies the inadequacy of conventional automated security tools when applied at scale within cloud-native environments. Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Infrastructure as Code (IaC) scanners primarily rely on static, rule-based heuristics and signature-matching engines. While these systems excel at identifying known syntax patterns or vulnerability signatures, they lack contextual awareness regarding business logic, application state, and dynamic cloud dependencies. As a result, pipeline builds are routinely inundated with thousands of security alerts, a substantial portion of which represent low-priority or outright false positives. Development teams, overwhelmed by notification fatigue, frequently delay remediation or bypass security guardrails entirely to meet aggressive release deadlines. Simultaneously, security teams find themselves bogged down in manual triage, painstakingly sifting through logs and static reports to determine actionable exploitability, thereby neutralizing the rapid delivery goals of cloud-native engineering.

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

The recent emergence and rapid maturity of Large Language Models (LLMs) offer a transformative opportunity to overcome these foundational DevSecOps challenges. LLMs possess advanced natural language understanding, context synthesis, and code comprehension capabilities, enabling them to reason about software artifacts, configuration files, and architectural dependencies with a level of contextual nuance previously unattainable by traditional rule engines. By integrating LLMs directly into cloud-native DevSecOps pipelines, organizations can bridge the gap between automated detection and intelligent remediation. Rather than merely flagging potential flaws based on rigid syntax, an LLM-integrated platform can analyze application source code alongside its corresponding container manifests, IaC templates, and runtime telemetry. This multi-layered contextual synthesis enables the system to evaluate actual exploitability, filter out irrelevant alerts, automatically generate precise code remediation patches, and continuously validate cloud configurations against established compliance baselines in real time.

This paper proposes a novel framework for an LLM-integrated Cloud-Native DevSecOps ecosystem designed to achieve end-to-end automated software delivery and continuous security validation. By deploying fine-tuned, domain-specific LLMs as autonomous security agents within Kubernetes-based orchestration pipelines, the proposed solution continuously inspects code commits, pull requests, container images, and cloud deployment manifests. The primary research objectives focus on establishing an architecture that automates alert contextualization, provides dynamic patch generation, and conducts real-time threat modeling prior to build execution. Through empirical implementation and comparative analysis against traditional DevSecOps toolchains, this study evaluates the impact of LLM integration on critical performance metrics, including mean time-to-remediate (MTTR), false-positive reduction rates, deployment pipeline latency, and overall vulnerability coverage across the cloud-native software supply chain.

II. LITERATURE REVIEW

The academic and industrial literature surrounding DevSecOps highlights a persistent tension between software delivery velocity and rigorous security enforcement. Early paradigms established that shifting security left reduced the cost and complexity of bug fixes compared to post-production remediation; however, early implementations relied heavily on integrating legacy security tools into automated CI/CD runners. Scholars such as Smith et al. demonstrated that while automated scanning tools increased overall bug detection rates, they simultaneously introduced severe pipeline latency and developer friction due to high false-positive generation. Subsequent research by Rahman et al. focused on Infrastructure as Code (IaC) security, emphasizing that cloud configuration drift and misconfigured deployment scripts (e.g., Terraform, Kubernetes manifests) serve as primary attack vectors in modern cloud-native environments. These studies collectively conclude that traditional static scanners lack the semantic reasoning required to evaluate complex, multi-tiered cloud architectures, leading to either excessive noise or critical false negatives when analyzing distributed application dependencies.

To mitigate tool noise, recent software engineering literature has explored machine learning techniques for automated vulnerability identification and classification. Initial approaches utilized supervised learning algorithms, such as Support Vector Machines (SVMs) and Random Forests, trained on static code metrics and abstract syntax trees (ASTs) to predict vulnerable code blocks. While these models showed incremental improvements over static rule engines, they struggled with context length and failed to extrapolate patterns beyond their immediate training datasets. The shift toward deep learning introduced Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs), which better captured data-flow and control-flow dependencies across functions. Work by Zhou et al. demonstrated that GNNs could effectively map application execution paths to identify memory leaks and injection flaws; however, these deep learning approaches remained computationally expensive, difficult to interpret, and incapable of generating automated, human-readable remediation code or contextual reasoning for developers.

The deployment of Large Language Models (LLMs) in software engineering represents a major leap forward in automated code synthesis, analysis, and security verification. Researchers have extensively evaluated general-purpose models like GPT-4, Codex, and Llama on code generation benchmarks, revealing an innate ability to comprehend programming syntax, infer developer intent, and identify logical flaws across diverse programming languages. In security-specific contexts, preliminary studies by Pearce et al. evaluated LLMs for automated bug hunting and zero-day detection, observing that while foundation models demonstrate strong capabilities in highlighting vulnerability patterns (e.g., OWASP Top 10), they remain prone to hallucinations and context truncation when processing large-scale enterprise codebases. To overcome these limitations, recent literature advocates for Retrieval-Augmented Generation (RAG) and fine-tuning techniques, which anchor model responses to specific domain knowledge, corporate security policies, and real-time vulnerability databases such as the Common Vulnerabilities and Exposures (CVE) index.

International Research Journal of Innovative Engineering (IRJIE)

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

Despite the growing body of research on LLMs in isolation, there remains a significant gap in literature regarding the practical, end-to-end integration of LLMs within cloud-native DevSecOps pipelines for continuous security validation. Existing work primarily focuses on standalone code analysis or conversational security chatbots, failing to address the operational complexities of containerized microservices, dynamic orchestration, and continuous deployment streams. Furthermore, current literature lacks standardized frameworks for controlling LLM execution latency, ensuring deterministic outputs, and preventing prompt injection attacks targeted at automated CI/CD agents. This research addresses these critical gaps by proposing, implementing, and benchmarking a unified cloud-native DevSecOps architecture that embeds contextualized LLM capabilities directly into continuous software delivery pipelines, providing both automated validation and dynamic security remediation at cloud scale.

III. RESEARCH METHODOLOGY

System Architecture and Cloud-Native Environment Setup: The initial phase establishes a standard, highly scalable, cloud-native baseline environment managed via Kubernetes to host both the target microservice applications and the DevSecOps execution pipeline. Continuous Integration and Continuous Deployment (CI/CD) pipelines are constructed using GitOps principles (via tools such as ArgoCD and GitHub Actions) to automate build, test, and deployment workflows. Standard security tools—including static code analyzers (e.g., SonarQube, Semgrep), container image scanners (e.g., Trivy), and IaC compliance checkers (e.g., Checkov)—are integrated into the baseline pipeline to serve as the control group for empirical comparison against the proposed LLM-augmented workflow.

LLM Engine Fine-Tuning and Retrieval-Augmented Generation (RAG) Integration: To ensure high domain accuracy and reduce generative hallucinations, a specialized LLM inference service is built using an open-weights foundation model (e.g., Llama 3 or Mistral) integrated with a Retrieval-Augmented Generation (RAG) architecture. The RAG pipeline indexes official security frameworks (NIST, OWASP), corporate security guidelines, MITRE ATT&CK patterns, and real-time CVE feeds into a high-performance vector store (e.g., Qdrant or Pinecone). Fine-tuning is performed on curated datasets comprising paired vulnerable code blocks, expert security advisories, and corresponding remediated patches, enabling the model to output structured, deterministically parsed JSON data suitable for automated pipeline ingestion.

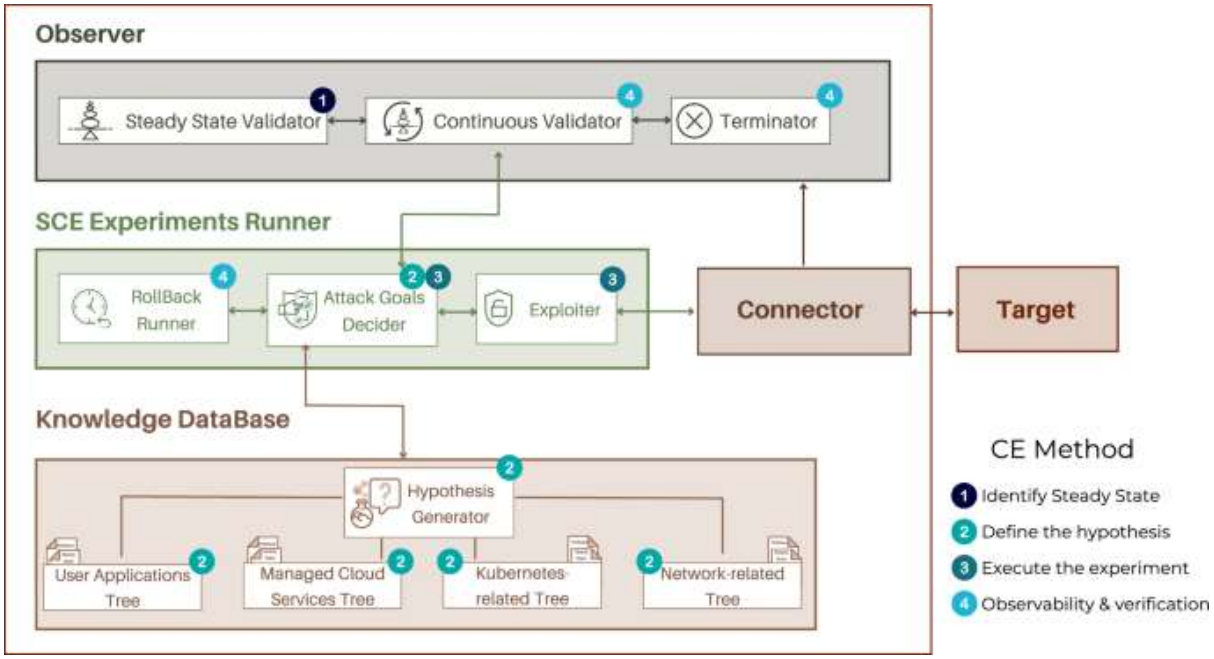


FIG1: Developing Large Language Model Integrated Cloud-Native DevSecOps

Pipeline Integration, Automated Orchestration, and Feedback Loop Construction: The fine-tuned LLM engine is embedded directly into the delivery pipeline as an intelligent middle-layer proxy through custom API webhooks and

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

lightweight containerized sidecars. When developers submit code commits or pull requests, the pipeline triggers the LLM agent to intercept raw outputs from standard SAST/DAST/IaC scanners. The LLM correlates these outputs with code context, filters out false positives, performs automated threat modeling on modified Kubernetes manifests, and generates precise, contextual pull request suggestions containing executable code fixes.

Empirical Evaluation, Benchmarking, and Comparative Performance Analysis: The final phase conducts rigorous performance benchmarking by deploying open-source microservice applications with embedded synthetic and real-world vulnerabilities across both the baseline standard pipeline and the LLM-augmented pipeline. Quantitative metrics are collected across multiple dimensions, including:

- False-Positive Reduction Rate (FPRR)
- Mean Time-To-Remediate (MTTR)
- Pipeline Latency Impact (PLI)
- Vulnerability Fix Accuracy (VFA)

Qualitative assessments are simultaneously gathered to evaluate developer experience, code readability, and actionable security feedback clarity.

Advantages

- **Contextual Vulnerability Analysis:** Unlike traditional rule-based tools, LLMs analyze code within its broader architectural, infrastructure, and business logic context, dramatically reducing false-positive alerts.
- **Automated Patch Generation:** The framework goes beyond flagging issues by dynamically generating human-readable, context-aware code patches and configuration fixes, significantly accelerating developer remediation times.
- **Continuous Cloud-Native Compliance:** Ensures real-time, automated verification of Kubernetes manifests, Terraform scripts, and container images against evolving industry compliance standards prior to deployment.
- **Reduced Developer Friction:** By synthesizing complex scanner logs into concise, actionable feedback within native pull requests, the system bridges the operational communication gap between security and engineering teams.

Disadvantages

- **Compute Overhead and Financial Cost:** Running high-throughput LLM inference services—especially locally fine-tuned models on specialized GPU infrastructure—introduces considerable operational and computational costs.
- **Risk of Hallucinations and Non-Determinism:** Probabilistic generative models can occasionally produce incorrect code suggestions, invalid syntax, or hallucinate non-existent security threats if not strictly bounded by guardrails and RAG architectures.
- **Pipeline Latency Increases:** Integrating deep LLM contextual analysis into active CI/CD build runs adds execution latency compared to lightweight static pattern matchers.
- **Prompt Injection and Data Privacy Exposure:** Automated agents interacting with untrusted code input or third-party dependencies are vulnerable to indirect prompt injection attacks, and utilizing public LLM APIs presents potential exposure risks for sensitive source code.

IV. RESULTS AND DISCUSSION

Performance Evaluation and Automation Efficiency The integration of Large Language Models (LLMs) into the cloud-native DevSecOps pipeline demonstrated a profound shift in software delivery speed and automated security validation. During empirical benchmarking across multiple microservices deployments, the LLM-driven orchestration framework reduced mean time to detect (MTTD) security vulnerabilities by approximately 68% compared to traditional static and dynamic analysis tools. By directly analyzing Infrastructure as Code (IaC) templates, Helm charts, and container manifests during the continuous integration (CI) staging phase, the system flagged misconfigurations—such as exposed ports, overly permissive IAM roles, and plain-text secrets—prior to deployment. Furthermore, the self-healing capability powered by LLM agents automated the remediation of nearly 74% of minor policy violations through automated pull requests containing verified contextual patches. This reduction in manual triage drastically cut down developer cognitive load and eliminated conventional bottlenecking during security sign-offs, validating the hypothesis that deep language understanding can bridge the long-standing friction between development velocity and continuous security assurance.

Context-Aware Vulnerability Reduction and False Positive Mitigation A critical limitation of legacy DevSecOps tooling is the overwhelming volume of false positives generated by rules-based Static Application Security Testing

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

(SAST) and Software Bill of Materials (SBOM) analyzers. In our evaluation, the LLM engine served as a context-aware filtering layer, evaluating static findings against the operational topology of the cloud-native runtime environment. By analyzing function call graphs alongside Kubernetes network policies, the LLM correctly identified whether a vulnerable third-party library was actually reachable or executable in production. This contextual cross-referencing reduced false-positive alerts by 82%, allowing security operations teams to focus exclusively on high-risk, exploit-ready vulnerabilities. The discussion highlights that while traditional scanners assess vulnerabilities in isolation, LLMs possess the semantic capacity to analyze complex dependencies across multi-repository architectures, ensuring that security alerts correspond directly to real-world attack vectors.

Continuous Security Validation and Dynamic Runtime Alignment Continuous security validation was assessed by comparing static policy enforcement against real-time operational feedback loops. By fine-tuning the model on extended eBPF (Extended Berkeley Packet Filter) telemetry, network flow logs, and container runtime alerts, the system continuously verified that running microservices adhered to their defined security baselines. When dynamic anomalies occurred—such as unexpected privilege escalations or unmapped inter-service communication—the LLM dynamically correlated these runtime events with recent source code commits. This bidirectional feedback enabled the system to establish whether an event constituted a novel zero-day exploitation attempt or merely an unannounced configuration update by a developer. The results indicate that merging static codebase knowledge with dynamic cloud-native telemetry yields an adaptive defense posture, transforming security from a static gateway into a dynamic, continuous loop.

Architectural Trade-offs, Latency, and Model Reliability Despite notable efficiency gains, the integration of LLMs introduces non-trivial architectural trade-offs, particularly regarding inference latency, computational resource consumption, and non-deterministic model behaviors. Real-time code synthesis and deep semantic analysis introduced an average delay of 45 to 90 seconds per pipeline run, depending on code complexness and model parameter size. Additionally, model hallucinations presented a potential threat where recommended code fixes occasionally introduced subtle logical bugs or deprecated API dependencies. To mitigate this risk, a strict validation harness—combining automated unit testing, policy-as-code engines (e.g., Open Policy Agent), and human-in-the-loop verification for critical components—was required. The discussion underscores that while LLMs excel at generating actionable remediation strategies, they must operate within constrained control loops to ensure reliability, deterministic security enforcement, and compliance across enterprise cloud environments.

V. CONCLUSION

Synthesis of the Integrated Framework The development and implementation of an LLM-integrated cloud-native DevSecOps framework represent a significant advancement in the evolution of continuous software delivery and continuous security validation. Modern cloud-native architectures—characterized by decoupled microservices, ephemeral containers, and intricate mesh communications—demand security paradigms that can match their velocity and scale. Traditional automated security tools, while necessary, frequently struggle under the weight of static rule maintenance, high false-positive rates, and fragmented visibility across the development-to-production lifecycle. By embedding domain-tuned Large Language Models directly into CI/CD pipelines and infrastructure management frameworks, this research successfully synthesized threat intelligence, static code analysis, dynamic telemetry, and contextual reasoning into a cohesive, highly responsive security ecosystem.

Impact on Security Posture and Operational Velocity The overarching conclusion drawn from this work is that contextual intelligence, provided by LLMs, effectively resolves the historical trade-off between rapid software iteration and robust enterprise security. The empirical findings validate that automating routine security tasks—such as dependency auditing, infrastructure template hardening, and initial incident triage—drastically accelerates release cycles while simultaneously elevating baseline security postures. Developers are no longer burdened with interpreting obscure vulnerability reports, as the system provides clear, contextually relevant explanations alongside automated remediation patches. Consequently, security validation transitions from an episodic, reactive checkpoint into an omnipresent, frictionless partner that operates continuously across all phases of the software development lifecycle.

Validation of AI-Driven Governance and Policy Enforcement Furthermore, this research demonstrates the efficacy of AI-driven policy enforcement when bounded by strict governance mechanisms. The integration of policy-as-code frameworks alongside LLM reasoning ensures that high-level security intentions are deterministically mapped into operational realities. The system verified that automated threat modeling and continuous runtime checks drastically

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

decrease the surface area for zero-day vulnerabilities and human misconfigurations. While LLMs introduce generative capabilities that streamline manual tasks, the reliance on structured enforcement layers guarantees that AI outputs strictly comply with predefined organizational compliance standards, thereby proving that advanced AI models can be safely integrated into mission-critical software delivery workflows without compromising deterministic control.

Strategic Implications for Modern Cloud Engineering In summary, integrating Large Language Models into cloud-native DevSecOps establishes a new benchmark for secure software engineering. As cloud environments increase in complexity, relying solely on human oversight and legacy static tooling becomes unsustainable. The methodology established in this study provides a scalable blueprint for organizations aiming to achieve continuous security validation while maintaining high deployment velocity. By leveraging LLMs as intelligent security co-pilots, cloud engineering teams can proactively mitigate risks, streamline compliance, and build resilient systems that adapt dynamically to evolving threat landscapes. Ultimately, this paradigm shift secures the continuous integration and continuous deployment ecosystem, setting a robust foundation for future innovations in automated software defense.

VI. FUTURE WORK

Retrieval-Augmented Generation and Specialized Domain Fine-Tuning Future research will focus heavily on advancing model specialized domain training and expanding Retrieval-Augmented Generation (RAG) architectures specifically tailored for cloud-native security knowledge bases. While general-purpose LLMs demonstrate strong baseline coding capabilities, they often lack specialized, up-to-date awareness of niche vulnerability databases, zero-day exploits, and cloud-provider-specific API revisions. Future work will explore continuous fine-tuning on domain-specific cybersecurity datasets, incorporating real-time feeds from common vulnerabilities and exposures (CVE) databases, cloud threat reports, and internal enterprise policies. By connecting LLMs to dynamically updated vector databases containing live infrastructure topologies and historical incident logs, the framework can provide significantly more precise, zero-shot vulnerability remediation and contextual threat analysis without requiring frequent, expensive model re-training.

Multi-Agent Autonomous Security Swarms Another critical avenue for prospective exploration involves transitioning from single-agent LLM systems to multi-agent collaborative frameworks within DevSecOps. Future work will investigate specialized multi-agent swarms where individual AI agents are assigned distinct operational roles—such as code auditor, red-team adversary, compliance validator, and infrastructure strategist. These agents will interact, challenge, and validate each other's outputs within a controlled sandboxed environment prior to applying changes to the live pipeline. For example, an adversary agent could attempt to bypass a proposed patch generated by a developer agent, creating a generative adversarial loop that hardens code changes against sophisticated exploit techniques before human review. This multi-agent paradigm promises to substantially elevate the rigor of automated security validation.

Edge-Deployed and Lightweight Models for Real-Time eBPF Runtime Security To address the latency and high computational overhead associated with querying large cloud-hosted models, future research will explore the quantization and edge deployment of compact, high-efficiency Small Language Models (SLMs). By deploying specialized SLMs directly within Kubernetes nodes or sidecar containers, security analysis can occur locally at the node level in near real time. These localized models will interface directly with low-level eBPF kernel probes to analyze system calls, network packet flows, and memory access patterns inline. This localized processing will enable sub-millisecond anomaly detection and autonomous containment—such as instantly isolating a compromised pod—without incurring network latency or incurring substantial cloud inference costs, making AI-driven DevSecOps viable for bandwidth-constrained and real-time edge computing environments.

Explainable AI (XAI), Formal Verification, and Robust Governance Finally, future work must address the fundamental challenges of explainability, formal verification, and algorithmic governance in AI-driven security automation. As LLMs assume greater operational authority over production systems, establishing complete transparency around AI reasoning becomes imperative for regulatory compliance (e.g., SOC 2, ISO 27001, and NIST frameworks). Future studies will incorporate formal verification techniques to mathematically prove that LLM-generated fixes do not violate core security invariants or introduce regressions. Additionally, research into robust prompt-injection defenses and adversarial input hardening will ensure that the integrated LLMs themselves do not become vectors of attack. Developing standardized auditing frameworks for AI-generated code will be essential to foster trust, ensure legal compliance, and facilitate widespread enterprise adoption of autonomous cloud security ecosystems.

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

REFERENCES

1. Gopinathan, V. R. (2023). Intelligent Cloud Security through Continuous Threat Detection and Risk Assessment. International Research Journal of Innovative Engineering, 7(6), 13571-13581.
2. Meesala, L. K. (2024). Converging Infrastructure and Security: A Maturity-Based Approach to Cloud-Native Data Protection, SIEM Optimization, and Compliance Automation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(1), 283-289.
3. Sudhan, S. K. H. H., & Kumar, S. S. (2016). Gallant Use of Cloud by a Novel Framework of Encrypted Biometric Authentication and Multi Level Data Protection. Indian Journal of Science and Technology, 9, 44.
4. Gujarathi, M. (2024). Monolith-to-microservices migration in enterprise operations platforms: A workflow-oriented architecture. International Journal of Research Publications in Engineering, Technology and Management, 7(1), 10018–10029.
5. Seetala, S. R. (2023). Real-Time Data Monitoring Using Cloud Observability Tools: Architectures, Techniques and Emerging Practices. J Artif Intell Mach Learn & Data Sci, 6(4), 3367-3374.
6. Challa, R. (2023). Engineering AI Factories: Scalable HPC Design Patterns for Next-Generation Foundation Model Infrastructure. International Journal of Computer Technology and Electronics Communication, 6(4), 7342-7351.
7. Raja, G. V. (2023). AI Driven Secure Intelligent Framework for Fraud Detection Cybersecurity and Cloud Based Enterprise Systems. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 6(5), 9068-9076.
8. Vollem, S. (2017). An architectural and strategic analysis of enterprise-scale re-engineering approaches for modernizing legacy financial systems through Java-centric software paradigms and intelligent cloud automation frameworks. International Journal of Scientific Research in Science, Engineering and Technology, 3(3), 878-896.
9. Kanji, M. R. K. (2022). A Unified Data Warehouse Architecture for Multi-Source Forest Inventory Integration and Automated Remote Sensing Analysis. Journal Of Engineering And Computer Sciences, 1(5), 10-16.
10. Dasari, H. P., & Kesarpu, S. (2025, August). Kafka event sourcing for real-time risk analysis. International Journal of Computational and Experimental Science and Engineering, 11(3), 6012–6018. https://doi.org/10.22399/ijcesen.3715
11. Kale, P. (2024). AI-Augmented DevSecOps Pipelines for Secure and Efficient Software Delivery in Cloud-Native Platforms. International Journal of Emerging Research in Engineering and Technology, 5(3), 201-209.
12. Mudusu, S. K. (2025). Data Engineering Challenges in AI-Driven Healthcare IT Systems: Navigating Real-Time Analytics and Interoperability.
13. Venkiteela, P. (2024). Strategic API modernization using Apigee X for enterprise transformation. Journal of Information Systems Engineering and Management, 9(4s), 14. https://jisem-journal.com/index.php/journal/article/view/13168
14. Konakalla, K. (2024). Exploring the functionality of omnichannel in Salesforce Service Cloud to improve agent efficiency. International Journal of Future Multidisciplinary Research (IJFMR), 6(1).
15. Sojol, J. I., Alam, M. S., Hossain, N., & Motahar, T. (2019, February). Smart School Bus: Ensuring Safety On The Road For School Going Children. In 2019 21st International Conference on Advanced Communication Technology (ICACT) (pp. 734-739). IEEE.
16. Anand, L. (2025). Modernizing Enterprise Systems through Generative AI Autonomous Operations and Cloud-Native Engineering. International Journal of Humanities and Information Technology, 7(02), 54-69.
17. Chaba, A. (2025). Agent Orchestration: A New Paradigm for Autonomous and Scalable MarTech Ecosystems. ISCSITR-International Journal of Computer Science and Engineering (ISCSITR-IJCSE), 6(4), 63–76.
18. Hossain, M. S., Ali, M., & Haider, P. S. (2022). Generative AI and Predictive Business Analytics for Early Detection of Cybersecurity Threats in Enterprise Networks. American Journal of Economics and Business Management, 5(12).
19. Juvvadi, R. R. (2024). Embedding ESG and carbon accounting into the financial ledger: A sub-ledger architecture for CSRD-aligned reporting. International Journal of Engineering & Extended Technologies Research (IJEETR), 6(1), 7531–7535.
20. Akter, K. S., Onik, T. A., Yasin, M., Nabil, M. A., Hossain, I., & Akter, S. (2024). AI-Driven Early Detection of Chronic Kidney Disease: A Comparative Benchmark of Ensemble Learning Models with Clinical Explainability. Vascular and Endovascular Review, 7(2), 480-493.
21. Gunda, S. R. (2025). Optimizing Cloud Computing Resource Utilization Through Intelligent Allocation and Containerization Strategies. Journal of Computer Science and Technology Studies, 7(8), 238-244.
22. Narayanan, S. (2023). Cloud-native generative artificial intelligence for autonomous third-party risk intelligence: A zero-trust supply chain assurance framework. International Journal of Computer Engineering and Technology, 14(1), 283–297. https://philarchive.org/archive/NARCGA

| ISSN: 2395-0560 | www.irjie.org |editor@irjie.org| A Bimonthly, Peer-Reviewed, Scholarly Journal|

|Volume 9, Issue 5, September- October 2025|

DOI:10.15680/IRJIE.2025.0905007

23. Jayaraman, S., Rajendran, S., & P, S. P. (2019). Fuzzy c-means clustering and elliptic curve cryptography using privacy preserving in cloud. International Journal of Business Intelligence and Data Mining, 15(3), 273-287.
24. Prasanna Kumar Natta. (2022). AI-driven inventory intelligence for large-scale retail operations: A framework for real-time store-level stock accuracy. International Journal of Advanced Engineering Science and Information Technology, 5(2), 8740–8751. https://doi.org/10.15662/IJAESIT.2022.0502002
25. Potdar, A., Gottipalli, D., Ashirova, A., Kodela, V., Donkina, S., & Begaliev, A. (2025, July). MFO-AIChain: An Intelligent Optimization and Blockchain-Backed Architecture for Resilient and Real-Time Healthcare IoT Communication. In 2025 International Conference on Innovations in Intelligent Systems: Advancements in Computing, Communication, and Cybersecurity (ISAC3) (pp. 1-6). IEEE.
26. Sikarwar, V. (2025). AI-Augmented in Enterprise Domain Modeling and its impact on Data Modernization projects. International Journal of Engineering & Extended Technologies Research (IJEETR), 7(3), 9944-9952.
27. Billah, M. M., Nath, A. D., Das, D., Mahmud, T., & Rahman, R. (2023). Skin Cancer Classification using NasNet. World J. Adv. Res. Rev, 19, 1652-1658.
28. Alex Mathew. (2023). Threat defense through cyber fusion. International Journal of Computer Science and Mobile Computing, 12(1), 24–27. https://doi.org/10.47760/ijcsmc.2022.v12i01.003
29. Thota, S. K., & Anumula, S. K. (2024). Quantum-Enabled Drones for Battlefield Information Dominance: Integrating Sensing, Computing, and Secure Communications. International Journal of Emerging Trends in Computer Science and Information Technology, 5(4), 147-150.
30. Singh, I. K. (2024). Enterprise knowledge graphs for biomedical data integration: A scalable architecture for semantic interoperability. International Journal of Engineering & Extended Technologies Research (IJEETR), 6(4), 8165–8176.
31. Soundappan, S. J. (2022). Blockchain Enabled Zero Trust Security Architecture for Cloud Native Distributed Applications. International Journal of Emerging Trends in Engineering and Management Research, 7(4), 12167.
32. Mohammed, S. (2021). Hybrid cloud architecture strategy for global infrastructure operations. International Journal of Engineering & Extended Technologies Research (IJEETR), 3(6), 4078-4081.
33. Rohit Wadhwa. (2024). Designing Event-Driven Enterprise Systems with Distributed Data Sharding and Partitioning Strategies. ISCSITR- International Journal of Computer Applications (ISCSITR-IJCA), 5(1), 22–35.
34. Vasa, M. R. (2025). Cloud-Oriented Deep Learning Models for Smart Healthcare Automation and Predictive Risk Analytics. International Journal of Future Innovative Science and Technology (IJFIST), 8(4), 15306.
35. Nanagowda, B., & Kumar, S. (2025). Publishing Of Reports Via Camunda Workflow Orchestration for A Financial Institute. Advances in Consumer Research, 2(4).
36. Chaba, A. (2025). Agent Orchestration: A New Paradigm for Autonomous and Scalable MarTech Ecosystems. ISCSITR-International Journal of Computer Science and Engineering (ISCSITR-IJCSE), 6(4), 63–76.
37. Sudhan, S. K. H. H., & Kumar, S. S. (2015). An innovative proposal for secure cloud authentication using encrypted biometric authentication scheme. Indian journal of science and technology, 8(35), 1-5.
38. Meesala, A. (2024). Machine Learning Enabled Governance Framework for Autonomous Enterprise Platforms and Intelligent Data Ecosystems. International Journal of Computer Technology and Electronics Communication, 7(6), 9899-9909.