

Architecture of AI-agent orchestration for automated testing and cyber threat analysis in DevSecOps environments

Emil Faure^{1,2,†}, Inna Rozlomii^{1,*,†}, Anna Kovalova^{3,†}, Bohdan Savchuk^{4,†} and Serhii Naumenko^{5,†}

¹Cherkasy State Technological University, Shevchenko Blvd., 460, Cherkasy, 18006, Ukraine

²State Scientific and Research Institute of Cybersecurity Technologies and Information Protection, M. Zaliznyaka Str., 3 (6), Kyiv, 03142, Ukraine

³National Technical University "Dnipro Polytechnic", Dmytro Yavornytskyi Ave., 19, Dnipro, 49005, Ukraine

⁴National Technical University "Igor Sikorsky Kyiv Polytechnic Institute", Beresteyskyi Ave., 37, Kyiv, 03056, Ukraine

⁵Bohdan Khmelnytsky National University of Cherkasy, Shevchenko Blvd., 81, Cherkasy, 18031, Ukraine

Abstract

The rapid growth of cloud-native infrastructures, continuous integration and continuous deployment pipelines, and distributed DevSecOps ecosystems has significantly increased the complexity of automated software testing and cybersecurity monitoring processes. Traditional QA systems and centralized orchestration approaches often demonstrate limited adaptability to dynamic deployment environments, high-frequency telemetry streams, and rapidly evolving cyber threats. This study proposes an architecture of AI-agent orchestration for automated testing and cyber threat analysis in DevSecOps environments based on coordinated interaction between intelligent agents and large language models. The proposed architecture integrates distributed AI agents responsible for automated testing, telemetry analysis, vulnerability detection, anomaly monitoring, and contextual decision-making within cloud-native CI/CD infrastructures. The methodology combines telemetry-driven anomaly detection, LLM-assisted code and infrastructure analysis, adaptive task prioritization, and asynchronous orchestration mechanisms for distributed cybersecurity operations. Experimental evaluation was conducted in a Kubernetes-based DevSecOps environment using Docker containerization, GitHub Actions pipelines, distributed telemetry collection mechanisms, and simulated cyberattack scenarios. The obtained results demonstrated improved threat detection accuracy, reduced orchestration latency, adaptive workload redistribution, and enhanced scalability compared to traditional QA pipelines and single-agent cybersecurity systems. The proposed orchestration architecture achieved a Threat Detection Rate of 96.8% while maintaining stable operation under dynamically changing cloud workloads and high-frequency telemetry conditions. The developed methodology can be applied to intelligent QA automation, adaptive cybersecurity monitoring, and distributed threat analysis in modern cloud-native DevSecOps infrastructures.

Keywords

DevSecOps, AI-agent orchestration, automated testing, cyber threat analysis, large language models, cloud-native infrastructures

1. Introduction

The rapid evolution of cloud-native infrastructures, CI/CD pipelines, and distributed microservice architectures has significantly transformed modern software engineering practices. The integration of development, security, and operational processes into unified DevSecOps environments enables accelerated software delivery while maintaining continuous monitoring and security assessment throughout the software lifecycle [1]. However, the growing complexity of containerized applications and dynamic

WCH'26: Fifth International Workshop on Cyber Hygiene, June 25–26, 2026, Krakow, Poland

*Corresponding author.

†These authors contributed equally.

✉ e.faure@chdtu.edu.ua (E. Faure); inna-roz@ukr.net (I. Rozlomii); Anna.kovalova@anbosoft.net (A. Kovalova); bogd.sav@gmail.com (B. Savchuk); sergey.naumenko94@vu.edu.ua (S. Naumenko)

ORCID 0000-0002-2046-481X (E. Faure); 0000-0001-5065-9004 (I. Rozlomii); 0009-0008-1434-7298 (A. Kovalova); 0009-0009-8360-6385 (B. Savchuk); 0000-0002-6337-1605 (S. Naumenko)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

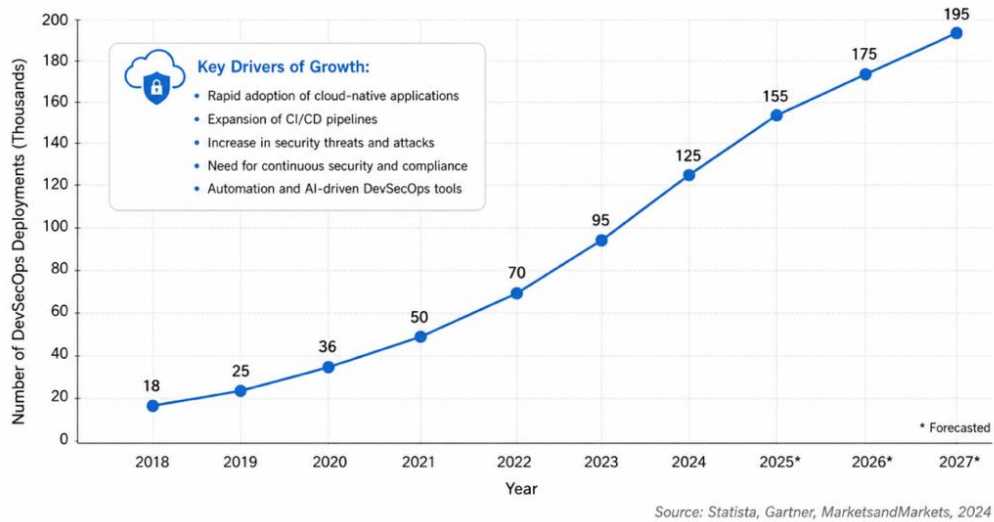


Figure 1: Growth of DevSecOps deployments in cloud-native infrastructures.

deployment pipelines expands the attack surface and introduces new cybersecurity challenges associated with software supply chain attacks, vulnerable dependencies, unauthorized access, and exploitation of cloud services.

The increasing adoption of DevSecOps methodologies has led to rapid growth of automated deployment pipelines and cloud-based testing infrastructures. Modern organizations rely on orchestration frameworks, container platforms, and intelligent monitoring systems to support scalable software delivery and vulnerability detection [2]. At the same time, traditional testing approaches based on isolated QA tools and manual threat analysis often demonstrate insufficient adaptability to rapidly evolving cyber threats and distributed deployment environments.

The global adoption of DevSecOps infrastructures continues to accelerate due to the growing demand for scalable and secure software ecosystems [3]. The increasing number of automated deployment pipelines and distributed software services demonstrates the necessity for intelligent orchestration mechanisms capable of coordinating testing processes and cyber threat analysis in real time [4]. The dynamics of DevSecOps deployment growth in enterprise cloud infrastructures are illustrated in Figure 1.

The increasing integration of automated testing, security verification, telemetry monitoring, and cloud orchestration components significantly increases the structural complexity of QA pipelines. Modern DevSecOps environments combine asynchronous services, distributed validation mechanisms, vulnerability scanners, container orchestration tools, and real-time telemetry analysis modules. Conventional single-agent or manually coordinated QA systems often fail to provide adaptive response mechanisms under high-load distributed conditions [5].

Large language models (LLMs) and multi-agent artificial intelligence systems are increasingly considered promising approaches for intelligent software testing, automated vulnerability analysis, and adaptive orchestration of distributed security processes [6]. AI-agent orchestration enables coordinated interaction between autonomous agents responsible for vulnerability detection, telemetry analysis, automated test generation, anomaly identification, and decision support within DevSecOps pipelines [7]. Such coordination supports intelligent prioritization of cyber threats and adaptive response generation based on contextual infrastructure analysis.

The increasing structural complexity of modern QA pipelines and interconnected DevSecOps components is illustrated in Figure 2.

Despite significant progress in AI-assisted cybersecurity and automated software testing, existing DevSecOps solutions still demonstrate limitations related to fragmented orchestration mechanisms, insufficient coordination between testing and security modules, and limited adaptability to dynamic cloud infrastructures. In addition, many systems lack intelligent multi-agent interaction models capable

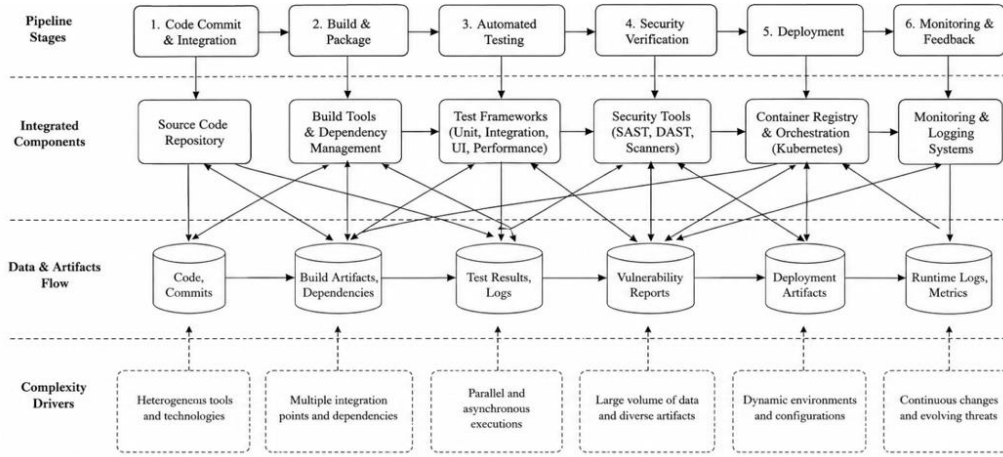


Figure 2: Structural complexity of modern QA pipelines in DevSecOps environments.

of performing coordinated cyber threat analysis in real time [8]. In addition, many traditional QA systems lack scalable mechanisms for integrating LLM-based reasoning and autonomous decision-making into distributed security workflows.

The aim of the study is to develop an architecture of AI-agent orchestration for automated testing and cyber threat analysis in DevSecOps environments based on coordinated interaction between intelligent agents and large language models.

2. Theoretical background and related works

The widespread adoption of cloud-native infrastructures and continuous software delivery models has significantly transformed modern software engineering processes. Contemporary DevSecOps environments integrate development, testing, deployment, monitoring, and cybersecurity verification into unified automated pipelines designed to support scalable and secure software delivery across distributed cloud infrastructures [9]. Modern DevSecOps frameworks typically combine container orchestration platforms, CI/CD services, telemetry monitoring systems, vulnerability scanners, and automated QA mechanisms operating in real time [10, 11].

Automated QA pipelines represent one of the core components of DevSecOps ecosystems [12]. Traditional software testing approaches based on sequential validation procedures and manually configured testing stages demonstrated limited scalability in distributed cloud environments [13]. The increasing complexity of microservice architectures and dynamic deployment infrastructures has expanded the number of dependencies and verification stages involved in software quality assurance. As a result, modern QA pipelines integrate automated testing, security analysis, telemetry monitoring, and runtime anomaly detection within continuous delivery workflows.

Existing orchestration approaches in DevSecOps environments are commonly based on workflow automation tools and centralized coordination mechanisms such as Kubernetes, GitHub Actions, GitLab CI/CD, and Jenkins pipelines. These platforms provide automated coordination of deployment and testing procedures; however, most orchestration mechanisms are primarily focused on infrastructure automation rather than intelligent cybersecurity analysis and adaptive testing coordination [14, 15, 16]. Conventional orchestration systems often demonstrate limited contextual awareness and insufficient capability for dynamic threat prioritization in rapidly evolving cloud environments [17].

The growing adoption of artificial intelligence technologies has stimulated the development of AI-assisted QA systems and intelligent cybersecurity frameworks. Multi-agent AI systems are increasingly considered effective solutions for distributed coordination of testing and security analysis tasks [17]. Autonomous agents can perform vulnerability scanning, telemetry analysis, anomaly detection, automated test generation, and incident prioritization while supporting adaptive orchestration and

Table 1

Comparison of Existing DevSecOps QA and Cybersecurity Approaches

Approach	AI support	Threat analysis	Multi-agent co-ordination	DevSecOps integration
Traditional QA pipelines	Limited	Partial	No	Partial
Rule-based security orchestration	Partial	Moderate	Limited	Moderate
Single-agent AI testing systems	Moderate	Moderate	No	Moderate
LLM-assisted QA frameworks	High	Moderate	Limited	Moderate
Multi-agent cybersecurity systems	High	High	High	Partial
Proposed AI-agent orchestration architecture	High	High	High	High

distributed workload management [18]. Compared to traditional single-agent architectures, multi-agent systems provide improved scalability, parallel task execution, and reduced orchestration bottlenecks in distributed infrastructures.

LLMs have recently become an important component of intelligent software engineering and cybersecurity research. LLM-based systems are increasingly applied in automated code review, vulnerability identification, software documentation analysis, and intelligent test generation [19]. Contextual reasoning capabilities of LLMs enable semantic analysis of source code, infrastructure logs, and telemetry streams, thereby improving automated QA procedures and cyber threat assessment.

AI-driven cyber threat analysis has also become an important research direction in cloud-native security monitoring. Modern cybersecurity systems integrate telemetry-based anomaly detection, behavioral analysis, and machine learning algorithms to identify suspicious activities in real time [20]. However, many existing monitoring systems still operate as isolated components without intelligent coordination between testing modules, cybersecurity agents, and orchestration mechanisms, which limits adaptive threat response capabilities in dynamic DevSecOps environments [21].

A comparative analysis of existing approaches related to AI-assisted QA systems, orchestration mechanisms, and cybersecurity analysis frameworks is presented in Table 1.

The analysis of related works demonstrates that existing approaches still reveal several limitations associated with insufficient coordination between automated testing and cybersecurity mechanisms, restricted scalability of centralized architectures, and limited integration of intelligent orchestration techniques within cloud-native DevSecOps infrastructures. These limitations confirm the necessity of developing adaptive AI-agent orchestration architectures capable of combining automated testing, cyber threat analysis, telemetry monitoring, and large language model reasoning within unified distributed DevSecOps environments.

3. Proposed AI-agent orchestration methodology

3.1. Architecture of AI-agent orchestration in DevSecOps environments

The proposed architecture supports intelligent orchestration of automated testing and cyber threat analysis in distributed DevSecOps environments. The framework combines specialized AI agents, telemetry monitoring mechanisms, cloud-native orchestration services, and large language model reasoning modules within a unified adaptive infrastructure. Its primary objective is to improve automated QA efficiency, reduce cyber threat response latency, and support adaptive coordination of security verification tasks in CI/CD pipelines.

The orchestration framework is based on a distributed multi-agent architecture that enables asynchronous task execution, adaptive workload redistribution, and event-driven coordination between cloud services. The general architecture of the proposed system is illustrated in Figure 3.

The proposed architecture includes the following core components:

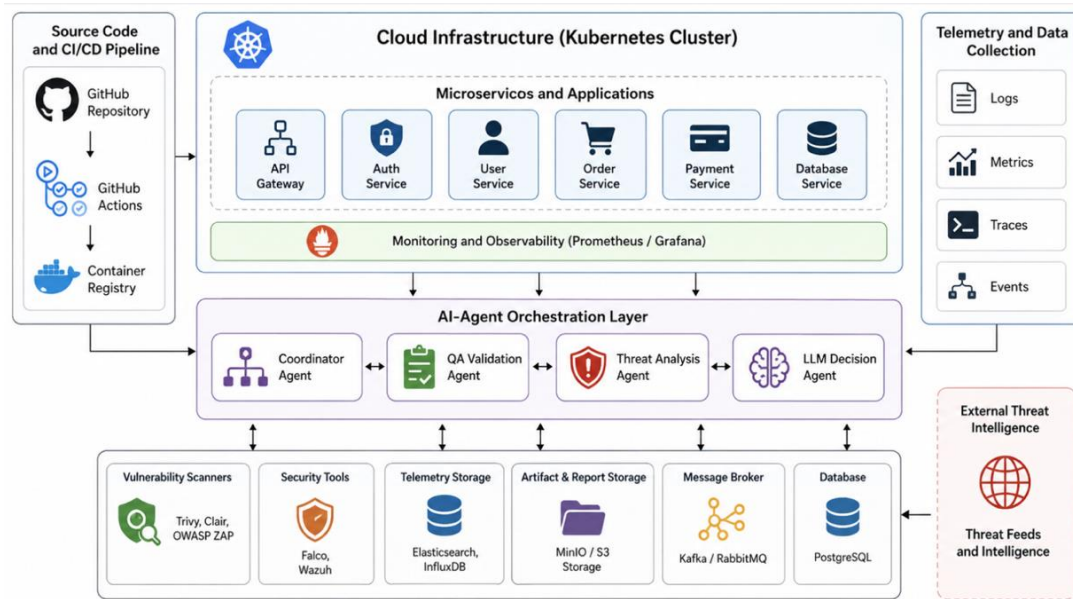


Figure 3: Architecture of AI-agent orchestration in DevSecOps environments.

1. Coordinator Agent – orchestration management, task scheduling, synchronization, and communication control between AI agents.
2. QA Validation Agent – automated unit testing, integration testing, API validation, regression analysis, and verification of deployment artifacts.
3. Threat Analysis Agent – vulnerability scanning, telemetry inspection, anomaly detection, and contextual risk prioritization.
4. LLM Decision Agent – semantic analysis of source code, telemetry streams, vulnerability reports, and generation of adaptive response recommendations.
5. Telemetry Monitoring Module – collection of runtime metrics, infrastructure logs, deployment events, and execution traces from distributed cloud services.
6. CI/CD Integration Layer – integration with Kubernetes clusters, Docker infrastructures, GitHub Actions, GitLab CI/CD pipelines, and cloud-native observability platforms.

The interaction between intelligent agents is organized according to an event-driven orchestration model. Operational events generated within the DevSecOps environment, including code commits, deployment activities, vulnerability detections, and telemetry anomalies, initiate asynchronous communication procedures between specialized AI agents. This coordination mechanism enables parallel execution of testing and cybersecurity analysis tasks without introducing centralized bottlenecks.

The orchestration workflow includes telemetry collection from distributed infrastructure components, event analysis and threat prioritization, and dynamic distribution of testing and cybersecurity tasks between AI agents. It also includes execution of automated QA and threat analysis procedures, aggregation of telemetry and vulnerability reports, and generation of adaptive recommendations and response actions by the LLM Decision Agent. The proposed architecture therefore supports adaptive orchestration, scalable telemetry analysis, and intelligent coordination of automated testing and cyber threat analysis in distributed cloud-native DevSecOps infrastructures.

3.2. Methodology of automated testing and cyber threat analysis

Adaptive orchestration of automated testing and cyber threat analysis is performed within distributed DevSecOps infrastructures characterized by dynamic workloads, asynchronous deployment activities, and heterogeneous cloud services. Unlike traditional QA systems based on sequential validation and static rule execution, the methodology utilizes coordinated interaction between specialized AI agents

capable of parallel analysis of software artifacts, telemetry streams, deployment events, and runtime anomalies [22]. Intelligent orchestration, contextual LLM reasoning, and telemetry-driven cybersecurity monitoring are integrated into a unified environment designed for continuous security verification and adaptive decision support.

The operational cycle begins with continuous acquisition of telemetry data generated by CI/CD pipelines, container orchestration services, runtime monitoring platforms, and distributed cloud applications. Infrastructure telemetry includes execution traces, deployment logs, API interactions, container metrics, authentication events, and resource utilization parameters. These telemetry streams are asynchronously transmitted to the orchestration environment where event correlation and distributed analysis are performed by AI agents.

Automated test generation is dynamically adapted according to infrastructure conditions and software modification history. Instead of predefined testing templates, the orchestration system generates contextual testing scenarios using semantic analysis of source code changes, deployment artifacts, historical vulnerability patterns, and runtime anomalies. Vulnerability scanning combines static analysis, telemetry inspection, dependency graph evaluation, and behavioral anomaly detection to identify suspicious execution patterns, unauthorized access attempts, abnormal container activities, and dependency-related attack vectors.

LLMs are integrated as contextual reasoning components responsible for semantic interpretation of software artifacts and operational telemetry. The LLM Decision Agent analyzes source code fragments, infrastructure logs, vulnerability reports, and deployment traces to identify hidden relationships between infrastructure events and cybersecurity incidents. LLM-assisted reasoning therefore enables detection of contextual anomalies and suspicious behavioral patterns that may remain undetected in conventional rule-based monitoring systems.

Telemetry-based anomaly detection utilizes temporal correlations between infrastructure metrics, deployment states, execution events, and communication patterns between distributed services. Instead of isolated threshold analysis, the methodology evaluates multidimensional telemetry trajectories describing the operational state of cloud-native infrastructures. The anomaly state vector for the monitored environment is represented as (1).

$$A(t) = \{M_c(t), L_e(t), N_f(t), D_s(t), R_u(t)\} \quad (1)$$

where $M_c(t)$ denotes container performance metrics; $L_e(t)$ represents log entropy variations; $N_f(t)$ corresponds to network flow anomalies; $D_s(t)$ describes deployment state transitions; $R_u(t)$ represents abnormal resource utilization dynamics.

The telemetry anomaly state is continuously evaluated through temporal event correlation between distributed infrastructure components. To estimate the propagation intensity of anomalous behavior across interconnected services, the orchestration system calculates the anomaly propagation function (2).

$$P_a(t) = \sum_{i=1}^n \frac{\Delta E_i(t) \cdot C_i(t)}{\tau_i} \quad (2)$$

where $P_a(t)$ represents anomaly propagation intensity; $\Delta E_i(t)$ denotes deviation of infrastructure events from normal operational behavior; $C_i(t)$ corresponds to service interaction connectivity; τ_i represents temporal synchronization delay between distributed components.

Threat prioritization within the orchestration environment is based on contextual infrastructure criticality and anomaly propagation dynamics rather than static vulnerability scores. The threat priority function is defined as (3).

$$T_p = \frac{\Phi_a \cdot S_c \cdot I_d}{\sqrt{R_t}} \quad (3)$$

where T_p represents threat priority; Φ_a denotes anomaly propagation density; S_c corresponds to service criticality; I_d represents infrastructure dependency depth; R_t denotes expected response time.

The proposed orchestration environment dynamically redistributes computational tasks between AI agents according to infrastructure state transitions and telemetry-derived threat conditions. Agent

coordination is performed using asynchronous event-driven communication mechanisms in which deployment events and anomaly detections initiate distributed task execution sequences. The orchestration process therefore eliminates centralized execution bottlenecks and supports parallel execution of cybersecurity analysis, telemetry processing, and software testing procedures.

To evaluate coordination efficiency of the distributed orchestration environment, the proposed methodology introduces an orchestration stability metric defined as (4).

$$O_s = \frac{\lambda_c}{\sigma_l \cdot w_f} \quad (4)$$

where O_s represents orchestration stability; λ_c denotes successful coordination throughput between AI agents; σ_l corresponds to orchestration latency variance; w_f represents frequency of coordination failures during distributed task execution.

Automated response triggering mechanisms are activated when telemetry propagation patterns indicate critical infrastructure instability or coordinated attack behavior. The orchestration system may dynamically initiate additional testing procedures, isolate suspicious containers, suspend unstable deployment processes, activate enhanced telemetry collection, or generate adaptive mitigation recommendations through LLM-assisted reasoning modules. Such adaptive response mechanisms enable reduction of cyber threat propagation time and improve resilience of distributed DevSecOps infrastructures under dynamically changing operational conditions.

The effectiveness of the proposed methodology is evaluated using operational metrics including Threat Detection Rate (TDR), Precision, Recall, F1-score, Orchestration Latency, anomaly propagation accuracy, and coordination stability. These metrics enable comprehensive assessment of automated testing efficiency, anomaly detection performance, adaptive orchestration capability, and resilience of the proposed AI-agent environment under distributed cloud-native operational conditions.

4. Experimental evaluation and performance analysis

4.1. Experimental environment and evaluation setup

Experimental evaluation was performed within a distributed cloud-native DevSecOps environment designed to emulate continuous integration, automated testing, and cybersecurity monitoring workflows. The infrastructure combined containerized deployment platforms, telemetry collection mechanisms, AI-assisted orchestration services, and vulnerability analysis tools operating under dynamic cloud conditions. The objective of the experimental setup was to evaluate intelligent QA orchestration and cyber threat analysis in distributed DevSecOps infrastructures.

The experimental environment was implemented using Docker-based containerization and Kubernetes orchestration services responsible for workload distribution and runtime management of cloud-native applications. Continuous integration and deployment operations were performed using GitHub Actions integrated with monitoring services and automated testing pipelines. The overall structure of the experimental environment is illustrated in Figure 4.

The cloud infrastructure consisted of distributed virtualized nodes hosting containerized microservices, telemetry processing modules, vulnerability scanning services, and AI-agent orchestration components. The deployed software ecosystem included REST-based cloud services, API gateways, authentication services, distributed databases, and simulated enterprise microservices communicating through asynchronous event-driven mechanisms.

To evaluate cybersecurity analysis capabilities, several simulated attack scenarios were executed, including unauthorized API access attempts, dependency injection attacks, privilege escalation activities, anomalous container behavior, insecure deployment configurations, and software supply chain manipulation scenarios. Telemetry collection was performed using distributed monitoring and observability services integrated into Kubernetes clusters and CI/CD pipelines. The collected telemetry streams included execution traces, deployment events, container metrics, infrastructure logs, API communication records, and resource utilization statistics used for anomaly detection and threat analysis.

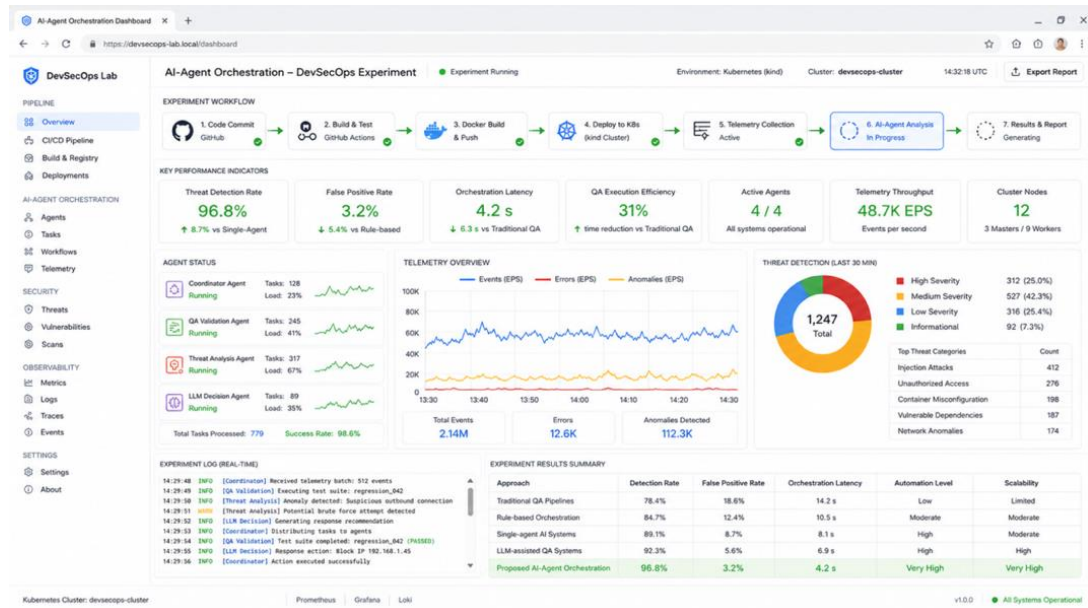


Figure 4: Experimental cloud-native DevSecOps infrastructure with AI-agent orchestration.

Table 2
Experimental Environment Configuration Parameters

Parameter	Value
Number of AI agents	4
Number of cloud nodes	12
Number of deployed microservices	28
Number of test cases	1450
Simulated attack scenarios	9
Telemetry streams	3200+ events/min
Containerized applications	18
Kubernetes clusters	3
Average deployment cycles per hour	24
Runtime monitoring interval	5 s

Datasets for vulnerability analysis were constructed using public cybersecurity repositories, simulated attack logs, software dependency vulnerability records, and synthetically generated telemetry anomalies. The main configuration parameters of the experimental environment are summarized in Table 2.

The experimental infrastructure was designed to emulate realistic operational conditions encountered in modern cloud-native DevSecOps environments characterized by high deployment frequency, distributed telemetry generation, asynchronous service interaction, and dynamically evolving cybersecurity threats. Such an environment enables comprehensive evaluation of orchestration scalability, adaptive threat analysis, telemetry-driven anomaly detection, and automated QA coordination mechanisms within distributed cloud infrastructures.

4.2. Results of performance and threat detection evaluation

The experimental evaluation demonstrated that the proposed AI-agent orchestration architecture significantly improves the efficiency of automated testing and cyber threat analysis in distributed DevSecOps environments compared to conventional QA pipelines and single-agent cybersecurity systems. The obtained results confirm that coordinated interaction between specialized AI agents enables reduction of response latency, improvement of threat detection accuracy, and increased scalability under dynamically changing cloud workloads.

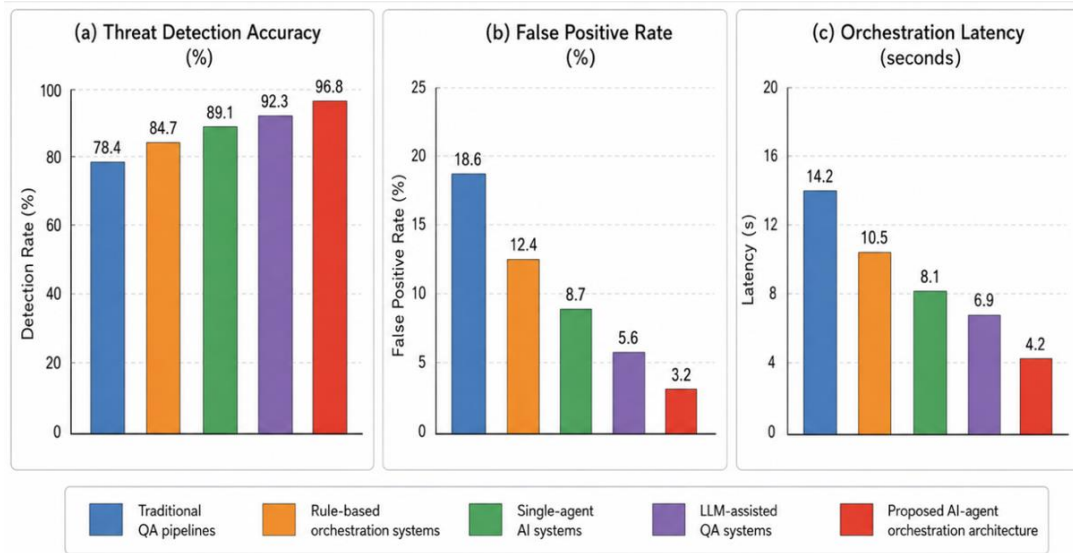


Figure 5: Comparative evaluation of threat detection accuracy, false positive rate, and orchestration latency.

The evaluation process focused on the following operational characteristics:

- threat detection accuracy;
- orchestration latency;
- adaptive response time;
- QA execution efficiency;
- cloud scalability;
- false positive rate;
- computational resource consumption.

The proposed orchestration environment demonstrated stable operation during simultaneous execution of distributed testing and cybersecurity analysis tasks across multiple Kubernetes clusters. During the experiments, the AI-agent coordination mechanism successfully processed high-frequency telemetry streams while maintaining low orchestration overhead and adaptive task redistribution between distributed agents.

One of the most important performance indicators was the effectiveness of cyber threat detection under simulated attack scenarios. The experimental results showed that the proposed architecture achieved high anomaly detection accuracy due to contextual telemetry analysis and coordinated interaction between the Threat Analysis Agent and the LLM Decision Agent.

The comparative performance of different testing and cybersecurity approaches is illustrated in Figure 5.

The obtained results demonstrated several important advantages of the proposed methodology:

- improved detection of anomalous infrastructure behavior;
- lower false positive rate;
- reduced orchestration delays during distributed task execution;
- adaptive redistribution of testing workloads;
- stable operation under increased cloud infrastructure load.

The proposed orchestration architecture achieved a Threat Detection Rate of 96.8%, which exceeded the performance of traditional QA systems and single-agent testing frameworks. The coordinated telemetry analysis mechanism also reduced the number of false positive detections during anomaly classification procedures.

Table 3
Performance Evaluation of Different QA and Cybersecurity Approaches

Approach	Detection Rate	Response Time	Automation Level	Scalability
Traditional QA pipelines	78.4%	14.2 s	Low	Limited
Rule-based orchestration systems	84.7%	10.5 s	Moderate	Moderate
Single-agent AI systems	89.1%	8.1 s	High	Moderate
LLM-assisted QA systems	92.3%	6.9 s	High	High
Proposed AI-agent orchestration architecture	96.8%	4.2 s	Very High	Very High

The evaluation metrics obtained during experimental execution are summarized in Table 3.

The proposed methodology demonstrated significant improvements in orchestration latency during parallel cybersecurity analysis and testing procedures. Due to asynchronous coordination between distributed agents, the orchestration environment maintained stable performance under increasing deployment frequency and telemetry volume.

Scalability analysis revealed stable operational efficiency under increasing numbers of cloud nodes and deployment events. During stress-testing scenarios, the orchestration system dynamically redistributed computational tasks between agents without introducing centralized bottlenecks, which is particularly important for large-scale cloud-native infrastructures with continuous deployment and high-frequency telemetry generation.

The experiments also showed that intelligent orchestration reduces QA execution time compared to traditional sequential testing pipelines. Dynamic prioritization of testing scenarios and adaptive telemetry analysis enabled selective execution of critical validation tasks without unnecessary computational overhead.

The average reduction of QA execution time reached:

- 31% compared to traditional QA pipelines;
- 22% compared to rule-based orchestration systems;
- 16% compared to single-agent AI testing architectures.

The resource consumption analysis showed that the proposed orchestration framework introduces moderate computational overhead associated with continuous telemetry analysis and LLM-assisted reasoning. Nevertheless, distributed execution of AI-agent tasks enabled efficient workload balancing between cloud nodes, thereby preventing critical resource saturation during large-scale deployment operations.

The relationship between telemetry processing intensity, orchestration latency, and resource utilization is illustrated in Figure 6.

The experimental evaluation confirmed that integration of multi-agent orchestration mechanisms with telemetry-driven cybersecurity analysis significantly improves operational resilience of DevSecOps infrastructures. The coordinated interaction between AI agents enabled adaptive response generation, accelerated vulnerability identification, and reduction of cybersecurity incident propagation within distributed cloud-native environments.

The obtained results demonstrate that the proposed AI-agent orchestration architecture provides:

- high threat detection efficiency;
- adaptive QA coordination;
- reduced response latency;
- scalable telemetry analysis;
- improved orchestration stability;
- effective operation under dynamic cloud workloads.

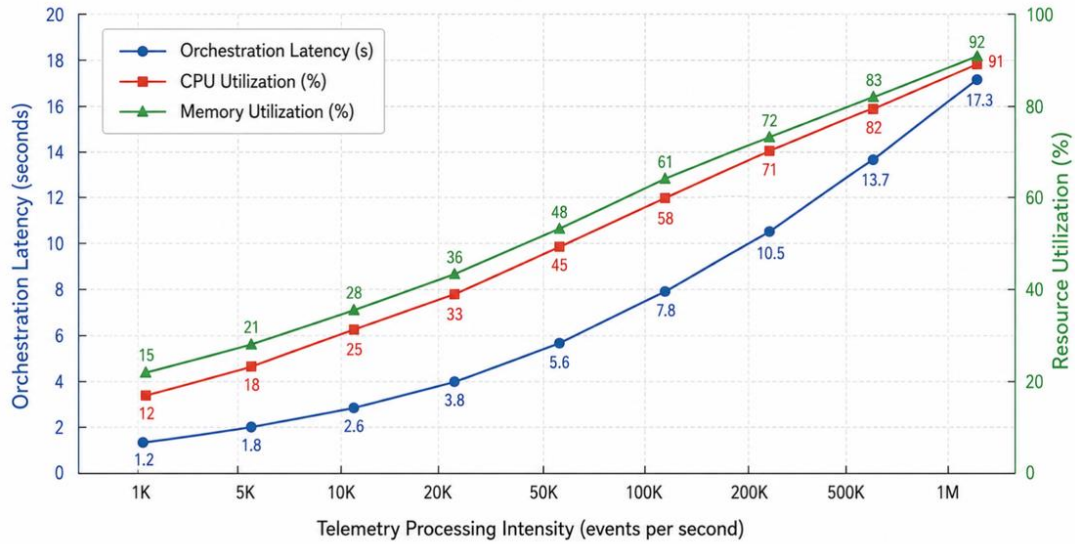


Figure 6: Relationship between telemetry workload, orchestration latency, and resource consumption.

These results confirm the practical applicability of the proposed methodology for intelligent software testing and cyber threat analysis in modern DevSecOps infrastructures characterized by distributed execution, continuous deployment, and high-frequency operational telemetry.

5. Discussion

The experimental results demonstrate that AI-agent orchestration provides significant advantages for automated testing and cyber threat analysis in distributed DevSecOps environments. Coordinated interaction between AI agents, telemetry-driven monitoring, and LLM-assisted reasoning enables adaptive orchestration of security verification procedures while reducing bottlenecks associated with centralized QA systems.

A major advantage of the architecture is its applicability to cloud-native ecosystems characterized by continuous deployment, distributed microservices, and high-frequency telemetry generation. The orchestration framework can be integrated into DevSecOps pipelines based on Kubernetes, Docker, GitHub Actions, GitLab CI/CD, and cloud observability platforms without significant modification of deployment workflows.

At the same time, several limitations remain important for large-scale deployment. Distributed telemetry synchronization, asynchronous communication between AI agents, and continuous infrastructure monitoring introduce additional orchestration overhead. Large-scale cloud infrastructures may also generate telemetry volumes exceeding processing capabilities of orchestration agents, requiring efficient workload balancing and optimization of telemetry analysis pipelines.

The reliability of LLM-assisted decision-making also requires careful consideration. Although large language models improve contextual interpretation of vulnerabilities and anomalies, semantic reasoning may occasionally generate incomplete or inconsistent threat assessments. Consequently, hybrid verification mechanisms combining AI reasoning with rule-based validation remain important for critical cybersecurity operations.

Additional security risks emerge from the autonomous nature of distributed AI agents. Manipulated telemetry streams or compromised orchestration agents may affect task prioritization and automated response generation. Therefore, secure authentication, encrypted telemetry exchange, and integrity verification mechanisms should be considered essential components of future orchestration environments.

Despite these limitations, the obtained results confirm the practical feasibility of intelligent AI-agent orchestration for adaptive software testing and cyber threat analysis in cloud-native infrastructures.

Future research directions include:

- self-healing DevSecOps infrastructures;
- federated AI-agent orchestration for multi-cloud environments;
- edge-cloud telemetry analysis for low-latency threat detection;
- reinforcement learning for orchestration optimization;
- autonomous vulnerability remediation mechanisms;
- explainable AI for transparent cybersecurity decision-making.

Future studies may also investigate integration of digital twin technologies and predictive infrastructure modeling for intelligent orchestration of large-scale DevSecOps ecosystems.

6. Conclusions

This study addressed the problem of adaptive automated testing and cyber threat analysis in modern DevSecOps environments characterized by distributed cloud infrastructures, continuous deployment pipelines, and dynamically changing operational conditions. The increasing complexity of cloud-native software ecosystems, high-frequency telemetry generation, and rapidly evolving cybersecurity threats create significant challenges for traditional QA systems and centralized orchestration mechanisms.

To address these challenges, an architecture of AI-agent orchestration for automated testing and cyber threat analysis in DevSecOps environments was proposed. The developed architecture integrates coordinated interaction between specialized AI agents, telemetry-driven monitoring mechanisms, large language model reasoning, and distributed orchestration services within a unified adaptive cybersecurity framework. The proposed methodology supports asynchronous task execution, intelligent QA coordination, telemetry-based anomaly detection, automated vulnerability analysis, and adaptive response generation in distributed cloud infrastructures.

The experimental evaluation confirmed the effectiveness of the proposed orchestration architecture under realistic DevSecOps operational conditions. The obtained results demonstrated improved threat detection accuracy, reduced orchestration latency, accelerated response time, adaptive workload redistribution, and scalable telemetry analysis compared to traditional QA pipelines and single-agent cybersecurity systems. The proposed environment achieved stable operation under increasing infrastructure load while maintaining efficient coordination between distributed AI agents.

The conducted experiments additionally demonstrated that integration of telemetry-driven monitoring with LLM-assisted reasoning significantly improves contextual interpretation of cybersecurity events and enables adaptive prioritization of threat analysis procedures. The proposed orchestration methodology also reduced QA execution time and improved operational resilience of distributed cloud-native infrastructures under simulated cyberattack scenarios.

The practical significance of the proposed architecture is associated with its applicability to modern DevSecOps ecosystems based on container orchestration platforms, CI/CD infrastructures, and distributed cloud services. The developed methodology may be integrated into enterprise software delivery pipelines to support intelligent QA automation, continuous cybersecurity monitoring, and adaptive orchestration of distributed testing procedures.

The main contributions of the study include:

- development of a distributed AI-agent orchestration architecture for DevSecOps environments;
- integration of LLM-assisted reasoning into automated testing and cybersecurity workflows;
- implementation of telemetry-driven anomaly detection and adaptive threat analysis;
- experimental evaluation of orchestration scalability and cybersecurity performance in cloud-native infrastructures.

The obtained results confirm that intelligent AI-agent orchestration represents a perspective direction for future development of adaptive DevSecOps infrastructures capable of supporting scalable automated testing, coordinated cybersecurity monitoring, and intelligent response generation in distributed cloud-native environments.

Acknowledgments

This research was funded by the Ministry of Education and Science of Ukraine under grant 0123U100270.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] N. K. K. R. Yelkoti, Security as code: An architectural framework for automated risk mitigation in devsecops pipelines, *Journal of Computer Science and Technology Studies* 7 (2025) 235–244. doi:10.32996/jcsts.2025.7.6.26.
- [2] Y. Voievodin, I. Rozlomii, Application security optimization in container orchestration systems through strategic scheduler decisions, in: *Proceedings of the CPITS-2024: Cybersecurity Providing in Information and Telecommunication Systems*, volume 3654 of *CEUR Workshop Proceedings*, 2024, pp. 471–478.
- [3] A. Yarmilko, I. Rozlomii, S. Naumenko, Dependability of embedded systems in the industrial internet of things: Information security and reliability of the communication cluster, in: *International Scientific-Practical Conference “Information Technology for Education, Science and Technics”*, Springer Nature Switzerland, Cham, 2024, pp. 235–249.
- [4] M. Aminu, A. Akinsanya, D. A. Dako, O. Oyedokun, Enhancing cyber threat detection through real-time threat intelligence and adaptive defense mechanisms, *International Journal of Computer Applications Technology and Research* 13 (2024) 11–27. doi:10.7753/IJCATR1308.1002.
- [5] W. Cong, W. Tao, B. Jinsong, Agentic digital twin-embedded maintenance methodology for energy equipment: A self-evolving operational paradigm, *Journal of Manufacturing Systems* 84 (2026) 100–116. doi:10.1016/j.jmsy.2025.11.020.
- [6] D. Mukherjee, A multi-agent security framework for ai-assisted software development, *International Journal of AI, BigData, Computational and Management Studies* 7 (2026) 32–38. doi:10.63282/3050-9416.IJAIBDCMS-V7I1P105.
- [7] O. Vakhula, I. Oprisky, Ai-driven security as code for software development using multi-agent systems, in: *CEUR Workshop Proceedings*, 2025. URL: <https://ceur-ws.org/Vol-4024/paper13.pdf>.
- [8] A. Yarmilko, S. Naumenko, Analysis of information security issues in balancing multiple independent containers on a single server, in: *Proceedings of the 3rd International Workshop on Information Technologies: Theoretical and Applied Problems*, 2023, pp. 450–461.
- [9] S. Talakola, Embedding cybersecurity into smart enterprise systems: An applied devsecops and automation approach, *International Journal of AI, BigData, Computational and Management Studies* (2026) 36–42. doi:10.63282/3050-9416.ICAIDSCT26-105.
- [10] A. Mittal, Ai-enabled devsecops for secure cloud deployments, in: *Revolutionizing the Cloud: Generative AI, Security, and Sustainability*, Springer Nature Switzerland, Cham, 2026, pp. 323–347. doi:10.1007/978-3-032-07479-9_16.
- [11] G. P. Rusum, Adaptive devsecops: Integrating ai-driven threat detection in continuous delivery pipelines, *American International Journal of Computer Science and Technology* 7 (2025) 13–27. doi:10.63282/3117-5481/AIJCST-V7I5P102.
- [12] S. Chittala, Securing devops pipelines: Automating security in devsecops frameworks, *Journal of Recent Trends in Computer Science and Engineering* 12 (2024) 31–44. doi:10.70589/JRTCSE.2024.5.5.
- [13] A. Dhandapani, Automation testing in microservices and cloud-native applications: Strategies and innovations, *Journal of Computer Science and Technology Studies* 7 (2025) 826–836. doi:10.32996/jcsts.2025.7.3.91.

- [14] S. T. Makani, S. Jangampeta, The evolution of cicd tools in devops from jenkins to github actions, *International Journal of Computer Engineering and Technology (IJCET)* 13 (2022) 166–174.
- [15] I. Kolawole, A. Fakokunde, Improving software development with continuous integration and deployment for agile devops in engineering practices, *International Journal of Computer Applications Technology and Research* 14 (2024) 25–39. doi:10.7753/IJCATR1401.1002.
- [16] R. Sannapureddy, S. Nelavelli, V. K. R. Kovvuri, Optimizing continuous integration and continuous deployment (ci/cd) pipelines: Strategies, tools, and performance metrics, *International Journal of AI, BigData, Computational and Management Studies* 2 (2021) 117–129. doi:10.63282/3050-9416.IJAIBDCMS-V2I4P112.
- [17] Y. V. Voievodin, I. O. Rozlomii, Advanced software framework for comparing balancing strategies in container orchestration systems, in: *Inderscience Proceedings*, 2024, pp. 60–69.
- [18] A. Mareedu, Autonomous security operations centers (soc): Ai agents for threat triage, response, and orchestration, *International Journal of Emerging Research in Engineering and Technology* 6 (2025) 63–70. doi:10.63282/3050-922X.IJERET-V6I2P108.
- [19] A. Ali, M. C. Ghanem, Beyond detection: Large language models and next-generation cybersecurity, *SHIFRA* (2025) 81–97. doi:10.70470/SHIFRA/2025/005.
- [20] S. Ghosh, S. Dadi, A comprehensive review of telemetry data-driven ai cybersecurity solutions in iot-based insurance ecosystems, *International Journal of Emerging Research in Engineering and Technology* 6 (2025) 103–112. doi:10.63282/3050-922X.IJERET-V6I3P112.
- [21] A. S. Adepoju, Adaptive program management strategies for ai-based cyber defense deployments in critical infrastructure and enterprise digital transformation initiatives, *International Journal of Research Publication and Reviews* 6 (2025) 5599–5615. doi:10.55248/gengpi.6.0725.2651.
- [22] R. Konda, Agentic ai for software development: Autonomous agents in requirements engineering, testing, and deployment, *International Journal of Emerging Research in Engineering and Technology* (2025) 139–148. doi:10.63282/3050-922X.AECTIC-118.