



Original Article

AI-Augmented DevSecOps Pipelines for Secure and Efficient Software Delivery in Cloud-Native Platforms

Pranay Kale

Automation Architect, Texas, USA.

Abstract - Cloud-native architectures have revolutionized software development, offering businesses the opportunity to achieve scalability, flexibility, and constant innovation. At the same time, distributed systems, microservices, containerized deployments and multi-cloud infrastructures have added a number of security challenges along the entire software development lifecycle (SDLC) process. The major focus of traditional DevOps practices is to speed up software delivery by leveraging automation and Continuous Integration/ Continuous Deployment (CI/CD) pipelines. As cyberattacks grew in number and complexity, however, security has become part of the SDLC and DevSecOps has entered the picture. While DevSecOps provides benefits, it can be a challenge for organizations to successfully integrate security into their deployment processes without slowing down the deployment process, causing the system to operate less efficiently or reducing developer productivity. Artificial Intelligence (AI) has become a game-changing technology that can help solve these problems by boosting automation, threat intelligence, and predictive security analytics. AI-enhanced DevSecOps pipelines integrate machine learning, intelligent orchestration, automated vulnerability scanning, anomaly detection, and predictive risk management to create secure and efficient software delivery pipelines in cloud-native environments. Incorporating AI-powered security tools into the CI/CD lifecycle enables organizations to preemptively detect vulnerabilities, focus resources on remediation, minimize false alarms and track system activity in real-time. In this study, the researchers explore how to design and implement AI-enhanced DevSecOps pipelines for cloud-native environments. The framework proposed will combine several capabilities, including AI-based static application security testing (SAST), dynamic application security testing (DAST), container image scanning, Infrastructure-as-Code (IaC) security validation, runtime threat intelligence, and automated compliance monitoring. Besides, the framework uses machine learning algorithms to study software artifacts, deployment patterns and operational telemetry data to detect potential security risks prior to production deployment. It explores the ways in which AI-powered automation can help to minimize vulnerability exposure, speed up deployment timelines, and enhance operational resilience. The results of their experiments show that AI-powered DevSecOps pipelines deliver a markedly higher velocity of accurate vulnerability detection, shorter mean time to remediation (MTTR), fewer deployment failures, and better compliance adherence than traditional DevSecOps approaches. The results show better security incident prevention, deployment efficiency and software quality. The findings highlight the need for embedding intelligent security mechanisms to provide security assurance and delivery agility in cloud-native CI/CD pipelines. The new model provides a flexible and scalable solution for enterprises aiming to achieve fast innovation while maintaining comprehensive cybersecurity in today's cloud-based environment.

Keywords - Artificial Intelligence, DevSecOps, Cloud-Native Computing, CI/CD Pipeline, Machine Learning, Software Security, Continuous Deployment, Cybersecurity Automation, Vulnerability Management, Container Security.

1. Introduction

1.1. Background and Motivation

Cloud-native computing has become the preferred approach to develop and deploy modern enterprise applications because of its scalability, flexibility and resilience [[1]]. Technologies like containerization, the use of Kubernetes for orchestration, serverless computing and microservices architecture have facilitated the efficient management of complex distributed systems, enabling fast application development and deployment. Meanwhile, software release cycles have undergone dramatic transformation, moving away from monthly or quarterly releases to continuous deployment practices that enable a number of updates throughout the day, resulting in greater agility and competitiveness. The DevOps philosophy was born to address these issues with the coordination of the dev and ops teams, focusing on automation, collaboration and continuous integration and delivery [[3]]. By eliminating silos between development and operations, DevOps was able to achieve better deployment speed and operational efficiency [[2]]. In many instances, however, security was dealt with separately, and as a kind of a late-after-thought, so potential vulnerabilities were identified late in the development process, causing delays, costs in remediation and risk in production environments. To address this constraint, DevSecOps was born as a spin-off of DevOps that incorporates security principles right into the CI/CD pipeline. This method ensures security is taken into account early in development as opposed to after. DevSecOps promises to provide secure software applications while maintaining the agility and efficiency of modern software delivery practices by integrating automated security testing, vulnerability scanning, and compliance checks across the software development lifecycle.

1.2. Importance of AI-Augmented DevSecOps Pipelines for Secure Software Development

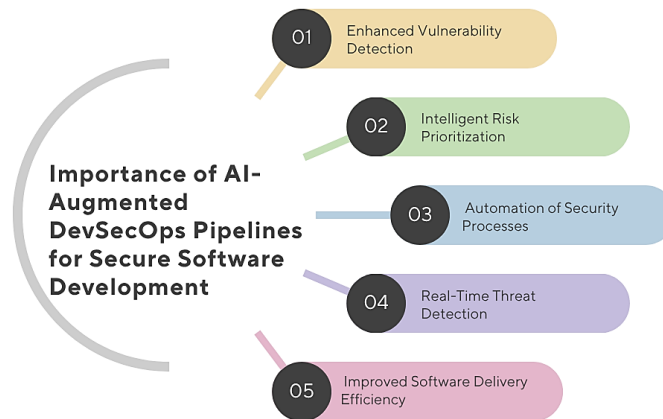


Fig 1: Importance of AI-Augmented DevSecOps Pipelines for Secure Software Development

1.2.1. Enhanced Vulnerability Detection

AI-enhanced DevSecOps pipelines greatly enhance the ability to identify vulnerabilities throughout code, dependencies, containers and infrastructure. Such machine learning models can detect intricate patterns and security vulnerabilities that rule-based models might not be able to detect. This results in quicker risk identification and minimisation of vulnerable code entering production environments [[4]].

1.2.2. Intelligent Risk Prioritization

Large software systems can contain thousands of vulnerabilities at the same time – prioritization can be hard to do manually. AI can overcome this challenge by calculating a changing risk score based on severity, exploitability, and business impact. This allows security teams to prioritize the most critical issues first, leading to increased efficiency and reduced alert fatigue [[6]].

1.2.3. Automation of Security Processes

AI-powered DevSecOps pipelines automate time-consuming and repetitive security activities like code scanning, dependency analysis, container inspection, and compliance testing. This automation decreases man power, helps lower errors, and helps ensure a secure policy is followed throughout the entire development lifecycle.

1.2.4. Real-Time Threat Detection

With AI, applications and infrastructure can be continuously monitored to identify anomalies and potential threats in real time. AI models can detect unusual behavior, network traffic, and runtime logs to quickly flag suspicious activity from these data analyses. This enables companies to react rapidly to security incidents and lessen potential harm [[5]].

1.2.5. Improved Software Delivery Efficiency

By leveraging AI throughout the DevSecOps pipeline, companies can strike a balance between robust security and efficient development processes. Automated decision making and predictive analytics decrease the times involved in deploying and remediation approaches. This means that software is sent out sooner with a high security level.

1.3. Problem Statement

Static rules, signature-based detection, and manual review processes are common traditional security testing methods within software development pipelines. These techniques work well for detecting known vulnerabilities but often result in a lot of security alerts, many of which are false alerts. This is a huge challenge for security and development teams because they have to spend a lot of their time and resources on analyzing and filtering down alerts that aren't a threat. With the increasing complexity of modern applications based on microservices, containerization, and 3rd party dependencies, the number of vulnerabilities found doubles, making the alert noise problem even worse. The high false positive rate can have significant implications, such as alert fatigue, where security teams can become overwhelmed with alerts and end up missing important threats. This compromises the security process as a whole and makes it more likely that actual threats will be overlooked [[7]]. Moreover, many of the traditional security methods are not context-aware, that is, they do not analyse vulnerabilities in isolation but take into account their exploitability, runtime environment and business impact. This can lead to sub-optimal prioritisation of remediation activities and can lack in alignment with the real level of risk in the organisation. One of the other factors is a balance between speedy software delivery and robust security. Modern DevOps environments focus on continuous integration and continuous deployment, with applications being deployed frequently and rapidly. Manual security validation however can add delays in the pipeline, especially when dealing with vulnerability triaging, impacting deployment cycles and

overall productivity. This poses a compromise between time and security which traditional systems have a difficult time achieving. It is critical, therefore, to have intelligent, automated and context-aware security mechanisms that minimize false positives, prioritize vulnerabilities based on actual risk and can easily integrate with DevOps development processes. This is a significant challenge that needs to be overcome for the secure, efficient, and scalable delivery of software in today's cloud-native landscape.

2. Literature Survey

2.1. Evolution of DevOps to DevSecOps

The transition from DevOps to DevSecOps was driven by organizations' understanding of the critical role security plays across the software development lifecycle. Most DevOps approaches focused on fast software delivery via continuous integration and continuous deployment (CI/CD) and security testing was typically performed as end-of-the-spectrum tasks. The result was often late detection of vulnerabilities, more expensive remediation and security risks. DevSecOps tackles these issues by integrating security measures, automated testing, compliance checks, and risk assessment seamlessly into the development and deployment process. DevSecOps allows organizations to deliver faster and have a better security position, since these security features are built in from the outset of software development, so that vulnerabilities can be detected and addressed before they end up in production environments [[8]].

2.2 Artificial Intelligence in Cybersecurity

Artificial Intelligence (AI) has emerged as a revolutionary technology in today's cybersecurity landscape, providing intelligent analysis of vast quantities of security data and automating the threat detection process. The use of AI-powered systems can lead to more precise and timely detection of suspicious patterns, anomaly detection, classification of malware, forecasting of cyber threats, and incident response activities, compared to traditional methods. Using machine learning algorithms, historical security events are used to train the system to better recognize new attack methods and threats as they appear. Random Forests, Support Vector Machines, Deep Neural Networks, Ensemble Learning, and Reinforcement Learning are among the techniques currently used for improving threat intelligence, intrusion detection, behavioral analytics, and prediction of risk [[9]]. With the evolution of cyberattacks, AI offers organizations proactive and adaptive security features, enhancing their defences [[10]].

2.3. AI-Driven CI/CD Security

AI has revolutionized the software delivery process by adding sophisticated tools and functions to CI/CD pipelines. AI has brought in a host of new tools and functions to the software delivery process, making it more secure and efficient. With AI-based CI/CD security solutions, tasks like code analysis, vulnerability prediction, security policy enforcement and deployment monitoring are automated, minimizing manual security reviews [[11]] [[12]]. By analyzing historical build data, source code repositories, and previous security incidents, AI models can identify patterns associated with vulnerabilities and predict potential security risks before deployment. Also, AI-driven anomaly detection systems track deployment actions and runtime performance to detect deviations that could signal security violations or configuration issues. These smart systems improve the productivity of DevSecOps processes by allowing quicker and more precise security decisions throughout the software development lifecycle [[13]].

2.4. Research Gaps

However, there are still several challenges in cybersecurity research that need to be addressed. While DevSecOps' and AI's strides in cybersecurity have been impressive, the following research challenges have yet to be solved: Most of the solutions available address only parts of the DevSecOps pipeline and don't offer security coverage across the entire software development lifecycle. In addition, existing vulnerability management methods are generally unable to implement adaptive prioritization, which can dynamically consider risk according to the changing threat landscape and organizational context. Many frameworks are also constrained in their ability to respond effectively to new attacks in the cloud-native and containerized space, due to runtime security intelligence. Moreover, there is a lack of empirical evidence on the effectiveness of AI-based DevSecOps solutions in large-scale enterprise use [[14]]. These constraints underscore the need for more unified, flexible, and intelligent security solutions that can overcome today's complex security issues in software ecosystems.

3. Methodology

3.1. Proposed AI-Augmented DevSecOps Architecture

3.1.1. Layer 1: Source Code Management

The Source Code Management layer is the bottom of the proposed framework, where source code is kept in secure and versioned repositories, typically with a Git-based platform. It contains branch protection rules, access rules, and secure coding rules to help maintain the integrity of code and prevent unauthorized changes. This layer provides a secure development environment and allows traceability throughout the software development lifecycle.

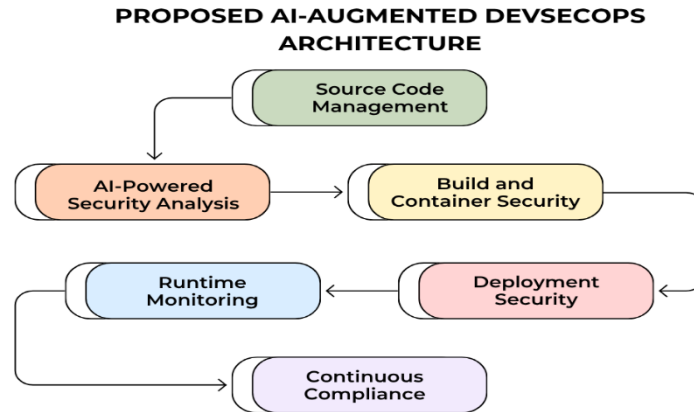


Fig 2: Proposed AI-Augmented DevSecOps Architecture

3.1.2. Layer 2: AI-Driven Security Analysis

The AI-Powered Security Analysis layer automatically analyzes the source code for vulnerabilities, coding flaws, and security misconfigurations, before application builds. It combines static code analysis, secret detection and dependency scanning tools with machine learning algorithms to spot potential risks. This layer helps to identify vulnerabilities early in the process, which can save a lot of remediation expenses and delays in development.

3.1.3. Layer 3: Build and Container Security

The Build and Container Security layer is concerned with the security of software artifacts and containerized applications during the build. It scans container images for known vulnerabilities and ranks vulnerabilities using AI to prioritise critical security issues. Also, supply chain security verification assures that third party libraries and dependencies are from trusted and verified sources.

3.1.4. Layer 4: Deployment Security

Deployment Security layer ensures that the security posture of applications is correct before they are deployed in to production environments. It includes Kubernetes policy validation, Infrastructure-as-Code (IaC) Analysis and configuration auditing to identify insecure configurations and policy breaches. This layer enables organizations to keep their deployment environment secure and compliant, and to reduce risks related to configuration.

3.1.5. Layer 5: Runtime Monitoring

The Runtime Monitoring layer monitors the behavior of applications and infrastructure in real-time after deployment and alerts to security threats. It leverages AI-driven threat detection, behavioral analytics, and security telemetry collection to recognize anomalies and suspicious activities. Regular monitoring helps to respond quickly to incidents and ensures the resilience of the system in the event of a cyber-attack.

3.1.6. Layer 6: Continuous Compliance

The Continuous Compliance layer guarantees that security controls are kept in sync with the organisation's security policies and legal obligations throughout the application lifecycle. It automates the auditing processes, maps security practices to relevant compliance standards and generates compliance scores for ongoing compliance evaluation. This preventive measure minimizes manual efforts for compliance and ensures organizations remain regulatory ready all the time.

3.2. Machine Learning-Based Risk Assessment

In the proposed framework, a Machine Learning-Based Risk Assessment module is added to intelligently assess and prioritize the discovered security vulnerabilities at different stages of the DevSecOps pipeline. Historically, vulnerability management has been mostly used to assign vulnerability severity ratings, which are not necessarily representative of the real risk of a vulnerability to an organisation. This can cause security teams to spend valuable time mitigating vulnerabilities that pose minimal business risk, and fail to consider vulnerabilities that can have a significant impact on business operations and/or financial losses. One solution to this is the proposed model which incorporates multiple factors, all of which contribute to the probability and possible impact of security threat. Three major parameters are used to calculate the risk score, Vulnerability Severity, Exploitability Score, and Business Impact. Vulnerability Severity is the technology's level of seriousness of a security vulnerability, usually based on a standard scoring system like the Common Vulnerability Scoring System (CVSS). Exploitability Score indicates the ease of an attacker exploiting the identified vulnerability given the attack complexity, privileges required, and the existence of exploit tools. Business Impact assesses the possible impact of a successful attack on the assets of the organisation: financial loss, disruption to operations, impact on reputation and regulatory implications. The

overall risk score in the proposed model will be a weighted sum of these three factors. The weight coefficients α , β , and γ represent the relative importance of the vulnerability severity, exploitability and business impact, respectively. These coefficients may be changed based on the priorities and security needs of the organization. For instance, in critical infrastructure, a greater weight may be given to business impact while in software development, a greater weight may be given to exploitability. This evaluation process is further improved by machine learning algorithms, which analyze past vulnerability information, security incidents, remediation results, and threat intelligence feeds. The system adapts the risk predictions over time and enhances their accuracy based on learned patterns to dynamically adjust predictions. This smart methodology allows security teams to prioritize most critical vulnerabilities first, optimize resources, ramp up remediation and ensure the overall security posture of the organization is improved and the risk of successful cyberattacks is minimized [[15]].

3.3. AI-Driven Threat Detection Model

The proposed AI-Driven Threat Detection Model aims to deliver real-time threat detection and intelligent identification of suspicious activities in today's cloud-native and DevSecOps environment. Conventional security monitoring systems are generally based on signatures and rules, which traditionally have not been effective in detecting threats that may be new or unknown, zero-day attacks and advanced techniques used by adversaries. To overcome these shortcoming, the proposed framework uses the machine learning based anomaly detection to get the capacity of detecting abnormal behaviors and security incident as real time [[16]]. The model works by continuously learning normal system behavior and comparing current operational activities to baseline normal behavior. If there are substantial differences from the normal behavior, the system will determine that the activity is an anomaly and alert security for subsequent investigation. The anomaly detection in this model is a binary decision method. When the model discovers abnormal behavior that it thinks is a potential security threat, it indicates that this behavior is abnormal. On the other hand, if the behavior observed is within the normal operating range, then nothing is done and no security action is taken. By doing so an automated detection of unknown attacks can be achieved that doesn't require explicit attack signatures and manually defined rules [[17]]. The machine learning engine is continuously analyzing several sources of operational and security data to further enhance the detection accuracy. The CPU utilization metrics are used to detect unusual patterns in the use of resources, which can be a sign of the occurrence of malware, cryptojacking, or denial of service attacks. The network traffic is analyzed for any suspicious communications, data exfiltrations, unusual connection behavior, or malicious network activity. Analyzing Container behavior to detect unexpected process execution, privilege escalation, unauthorized file modification, and unusual run-time actions in containerized environments. Furthermore, API activity logs are continually monitored for unusual API access behavior, unauthorized requests, excessive API usage, and possible application-layer attacks. The more telemetry data that is processed the more the machine learning model learns about normal and abnormal operation of the machine. This feature of adaptive learning helps minimize false alarms, increase detection accuracy, and allow for the quick identification of new threats [[18]]. As a result, the AI-based threat detection model further enhances the security of organizations by ensuring proactive, real-time protection and rapid incident response throughout the DevSecOps continuum.

3.4. Workflow Implementation

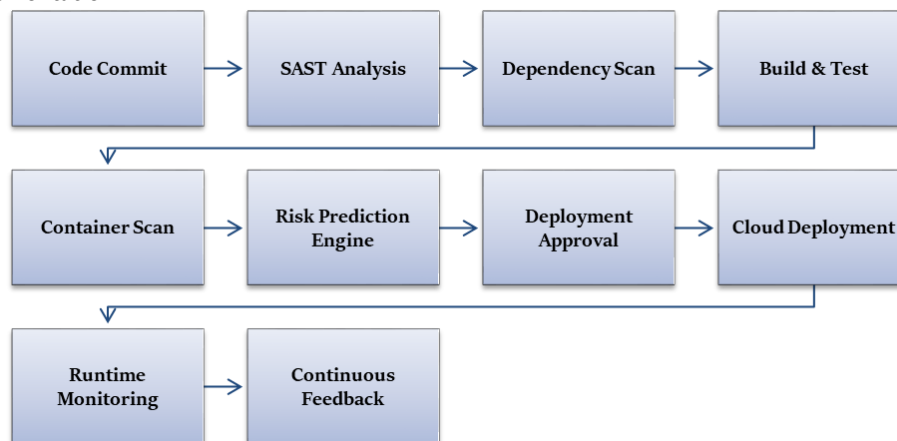


Fig 3: Workflow Implementation

3.4.1. Code Commit

The process starts with the developers pushing source code changes into a version control repository. Automated triggers start security and quality assessment process, allowing all code changes to be assessed before moving along the pipeline. This step creates traceability and enables safe software development methods.

3.4.2. SAST Analysis

Static Application Security Testing (SAST) is conducted at the time of code submission, looking for vulnerabilities in the source code. The analysis identifies coding vulnerabilities, insecure coding methodologies and possible security issues without

running the application. Early detection enables developers to take corrective measures before the problems can spread to the subsequent phase.

3.4.3. Dependency Scan

The third-party libraries, frameworks, and packages that the application uses are analyzed during the dependency scanning stage. It detects vulnerable software, old software, and insecure software dependencies that might pose security threats. It improves software supply chain security and decreases the vulnerability to external threats.

3.4.4. Build & Test

The application is compiled and automatically tested during build and test phase. Functional tests confirm that the software performs the necessary functions and functions correctly, while integration tests confirm that the software can be integrated into the system without any problems, and security tests confirm that the software does not violate the security standards required for the system. Successful completion will be readiness for packaging and deployment preparation.

3.4.5. Container Scan

After packaging into an image, automated scanning tools check the image for vulnerabilities and configuration flaws. The process examines the packages, application requirements and security configurations of the operating system in the container. This way, only safe and compliant images will advance through the pipeline.

3.4.6. Risk Prediction Engine

The Risk Prediction Engine uses machine learning algorithms to process data on vulnerabilities, security findings, and operational circumstances. It calculates the risk score and prioritizes and sorts issues based on their severity, exploitability, and business impact factors. This allows for better decision making and more effective use of remediation resources.

3.4.7. Deployment Approval

The system analyzes the security assessment results and risk scores prior to deployment, and compares them to pre-defined policies for the organization. Applications that conform to security requirements are automatically approved, and high-risk deployments might be subject to further review. This stage guarantees that only secure releases are put into production environments.

3.4.8. Cloud Deployment

Automated deployment mechanisms are used to deploy approved applications to cloud infrastructure. Consistency and compliance are maintained by security controls, configuration policies and deployment standards throughout the process. Automation minimizes deployment mistakes and quickly delivers software.

3.4.9. Runtime Monitoring

Once deployed, runtime monitoring keeps an eye on activities occurring in application and infrastructure, in real time. Unauthorized use, deviations from normal operations or policy violations are identified by analyzing security telemetry, behavioral patterns and operational metrics. This proactive monitoring enables early detection and response to threats.

3.4.10. Continuous Feedback

The last stage offers ongoing feedback to development, security and operations teams, leveraging monitoring results and security insights. Our lessons learned from incidents, vulnerabilities, and performance metrics are made part of future development cycles. This feedback loop helps to drive continuous improvement and enhance the overall DevSecOps process.

4. Results and Discussion

4.1. Experimental Setup

The proposed framework of AI-enhanced DevSecOps was tested in a cloud-based setting with the goal of mimicking contemporary software development and deployment workflows [Error! Reference source not found.]. The main orchestration platform was a Kubernetes cluster, which facilitated the management of containerized applications and ensured scalability, reliability, and automated resource management. Jenkins was used to automate code integration, testing, security validation and deployment workflows as Continuous Integration and Continuous Deployment (CI/CD) platform. Docker containers were used to ensure applications and their requirements are packaged into lightweight, portable and consistent execution environments. This configuration allowed the framework to mimic common DevSecOps operations in the real world, typical of enterprise organizations. SonarQube was added to the CI/CD pipeline to ensure secure development lifecycle, by performing Static Code Analysis and Source Code quality assessment. SonarQube identified coding vulnerabilities, security hotspots, code smells and maintainability issues as soon as they were introduced in the code base. Trivy Scanner was integrated so as to conduct in-depth vulnerability scanning of container images, OS packages, and third-party dependencies. This enabled the detection of well-known security issues prior to the deployment of applications in production environments. Furthermore, a Machine Learning Risk Engine has been created and embedded in the pipeline to apply machine learning to

security results that come from different security scanning tools. It was used to assess vulnerability severity, exploitability characteristics, historical attack data and the business impact factors to create dynamic risk scores and prioritize remediation efforts. Several key performance indicators (KPIs) were used to assess the effectiveness of the proposed framework. The framework was assessed for its accuracy in detecting vulnerabilities in source code, dependencies and container environments with Vulnerability Detection Rate. The Deployment Success Rate was the percentage of successful deployments that also met pre-defined security requirements. The Mean Time to Remediation (MTTR) metric is calculated to determine the average time needed to identify, prioritize, and resolve vulnerabilities, offering insight into the efficiency of security operations. Compliance Accuracy assessed the framework's ability to monitor and audit the organization automatically and continuously to meet policies and regulations. Together, these assessment metrics gave a full picture of the framework's security effectiveness, operating efficiency, and its ability to be implemented in today's cloud-native DevSecOps environments.

4.2. Performance Evaluation

Table 1: Performance Evaluation

Metric	Traditional DevSecOps (%)	AI-Augmented DevSecOps (%)
Vulnerability Detection Accuracy	82	96
False Positive Reduction	65	92
Deployment Success Rate	88	97
Compliance Accuracy	84	98
Threat Detection Efficiency	79	95
Mean Time to Remediation	70	93

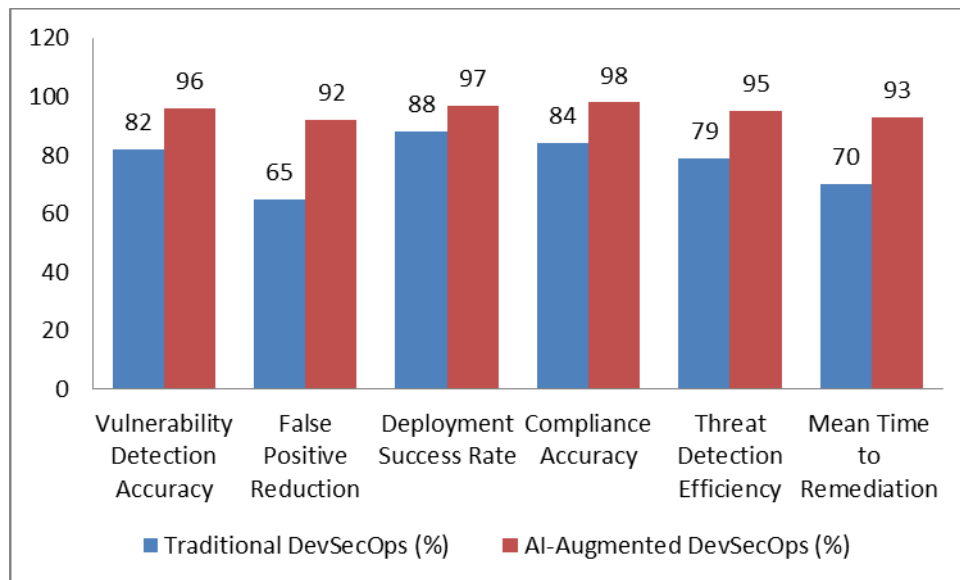


Fig 4: Performance Evaluation

4.2.1. Vulnerability Detection Accuracy

The proposed AI-Augmented DevSecOps framework was able to detect vulnerabilities with a 96% accuracy rate, compared to the traditional DevSecOps approach, which detected vulnerabilities at 82%. The improvement is largely due to incorporating machine learning algorithms that can detect intricate security patterns and undiscovered weaknesses. This improved detection ability minimizes the risk of security issues being present in production.

4.2.2. False Positive Reduction

In the traditional DevSecOps environment, the false positive reduction was 65%, while in the AI-augmented environment, it was increased to 92%. Intelligent risk assessment and anomaly analysis led to a reduction in unnecessary investigation of non-critical alerts. This made it possible for security teams to more efficiently direct their resources to mitigating real vulnerabilities.

4.2.3. Deployment Success Rate

The deployment success rate increased from 88% to 97% with the implementation of AI-driven security validation mechanisms. Automated security checks, risk-based approval processes and ongoing compliance verification ensured only secure and stable releases moved through the deployment pipeline. This enhancement added to the reliability of operations and minimizes deployment failures.

4.2.4. Compliance Accuracy

The results showed that the framework achieved a 98% compliance accuracy, whereas the traditional DevSecOps practices achieved 84% accuracy. Consistent compliance with security standards and organizational needs was achieved through automated auditing, regulatory mapping and ongoing policy enforcement. This function greatly decreased compliance errors and manual auditing tasks.

4.2.5. Threat Detection Efficiency

In a conventional environment, the threat detection efficiency was 79%, whereas it rose to 95% in the AI-assisted setting. The system could quickly detect suspicious activities and new threats thanks to continuous monitoring and machine learning-based behavioural analytics. The proactive detection feature enhanced the organisation's security stance and incident response efficiency.

4.2.6. Mean Time to Remediation Improvement

The proposed framework's remediation improvement rate was 93%, whereas it was 70% with traditional DevSecOps implementations. Security teams prioritized vulnerabilities using AI-driven methods and automated risk scoring to pinpoint and address critical vulnerabilities more efficiently. This led to a substantial decrease in the average time taken to remediate vulnerabilities, resulting in improved operational efficiency and security resilience.

4.3. Discussion

The findings of the experiments clearly illustrate the substantial improvements of the proposed AI-Augmented DevSecOps framework compared to the traditional DevSecOps approaches in terms of security and operation. A major benefit is the ability of AI to prioritize vulnerabilities, minimizing alert fatigue by sifting through and ranking security incidents by severity, exploitability, and business impact. This helps to prevent low priority alerts coming in to the security team and allows them to focus their time on any vulnerabilities that are most likely to impact the organisation. The overall security efficiency and responses are significantly enhanced as a result. Another desirable result is to improve run-time anomaly detection. By continuously monitoring system behavior and analyzing real-time telemetry data, the framework is able to quickly identify suspicious activities and potential security breaches. This proactive detection capability minimises the detection window for emerging threats and ensures faster incident response, thus enhancing the resilience of cloud-native applications and infrastructure. The use of machine learning models also enhances accuracy in detection, adapting to new attack patterns and minimizing false positives. Results also demonstrate significant improvements in deployment success rates, likely due to implementation of proactive risk assessment mechanisms in advance of production releases. The framework reduces the risk of vulnerabilities, configuration issues, and compliance violations by evaluating them beforehand and ensuring only secure and validated applications are deployed. This will minimise deployment failures and improve the stability of the system in production environments. Moreover, ongoing compliance monitoring is essential to staying up to date with regulatory requirements without the need of in-depth manual audits. Automated policy enforcement and real-time compliance monitoring provide complete application compliance to organizational and legal specifications at all stages of the application development lifecycle. This not only helps to minimize operational overheads but also enhances audit preparedness. In summary, the proposed framework succeeds in balancing the security assurance and deployment agility. This shows how AI can greatly improve security intelligence, operational efficiency, and scalability within DevSecOps pipelines, especially in today's cloud-native landscape that prioritizes speedy deployment and comprehensive security [[19]].

5. Conclusion

Many software systems are now de facto cloud-native, built using microservices and containers, and have come a long way from the traditional approach to software development. At the same time, the adoption of cloud-native platforms, microservices architectures, and containers has changed the way software is built, adding complexity and attack surface. In this dynamic environment, security must be built into the software development lifecycle (SDLC) and is no longer a feature one can add on. Although DevSecOps has helped with this shift by integrating security practices into Continuous Integration and Continuous Deployment (CI/CD) pipelines, traditional DevSecOps methods can often fall short of handling the scale, speed, and dynamism of today's cloud-based environments. These constraints underscore the critical need for more intelligent, adaptive, and automated security mechanisms that can adapt to and react to changing cyber threats in real time.

In this paper, the author introduced an AI-Augmented DevSecOps architecture that used AI and machine learning throughout various phases of the software delivery process to solve the identified challenges. The suggested architecture integrates smart features into source code analysis, dependency scan, container scan, infrastructure-as-code validation, runtime monitoring, and continuous compliance assessment to augment typical DevSecOps workflows. The framework also includes machine learning models that allow for vulnerability prioritization, anomaly detection, and predictive risk assessment, helping to identify security vulnerabilities more accurately and prioritize critical vulnerabilities for action. This will minimize the manual workload, facilitate quicker decision making time, and increase the effectiveness of the security process.

The experimental assessment of the proposed framework showed promising results on all the metrics, such as detection of vulnerabilities, compliance verification, deployment reliability, threat detection, and mean time to remediation. Overall, the findings highlight the potential of AI-powered automation in DevSecOps pipelines to enhance the security of software systems while also boosting their performance and efficiency. In conclusion, these findings underscore the benefits of integrating AI-driven automation into DevSecOps pipelines, not only for the security of software systems but also for their performance and efficiency. Moreover, the framework's smart analytics features help to make quicker and more knowledgeable choices, paving the way for more effective allocation of resources and enhanced system resilience.

The results of this study indicate that artificial intelligence is becoming a key strategic enabler in next generation DevSecOps practices. With AI-enhanced security pipelines, organizations can attain a balance of strong security, operational efficiency, and quick software delivery. This is especially well suited for the modern cloud-native environment where continuous deployment and robust security assurance go hand in hand.

References

- [1] Myrbakken, H., & Colomo-Palacios, R. (2017). DevSecOps: A multivocal literature review. In *Software Process Improvement and Capability Determination* (pp. 17–29). Springer. https://doi.org/10.1007/978-3-319-67383-7_2
- [2] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
- [3] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., & Wang, C. (2018). Machine learning and deep learning methods for cybersecurity. *IEEE Access*, 6, 35365–35381. <https://doi.org/10.1109/ACCESS.2018.2836950>
- [4] Conti, M., Dehghantaha, A., Franke, K., & Watson, S. (2018). Internet of Things security and forensics: Challenges and opportunities. *Future Generation Computer Systems*, 78, 544–546. <https://doi.org/10.1016/j.future.2017.07.060>
- [5] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19–31. <https://doi.org/10.1016/j.jnca.2015.11.016>
- [6] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *Proceedings of the IEEE Symposium on Security and Privacy*, 305–316. <https://doi.org/10.1109/SP.2010.25>
- [7] Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3), 160. <https://doi.org/10.1007/s42979-021-00592-x>
- [8] Wiafe, I., Koranteng, F. N., Obeng, E. N., Assyne, N., Wiafe, A., & Gulliver, S. R. (2020). Artificial intelligence for cybersecurity: A systematic mapping of literature. *IEEE Access*, 8, 146598–146612. <https://doi.org/10.1109/ACCESS.2020.3013145>
- [9] Alauthman, M., Aslam, N., Alashhab, Z., Al-Qerem, A., & Al-Omari, E. (2022). Machine learning for cybersecurity applications. *Electronics*, 11(3), 395. <https://doi.org/10.3390/electronics11030395>
- [10] Taddeo, M., & Floridi, L. (2018). How AI can be a force for good in cybersecurity. *Science*, 361(6404), 751–752. <https://doi.org/10.1126/science.aat5991>
- [11] Shu, R., Gu, X., & Enck, W. (2017). A study of security vulnerabilities on Docker Hub. *Proceedings of the ACM CODASPY Conference*, 269–280. <https://doi.org/10.1145/3029806.3029832>
- [12] Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1), 14–76. <https://doi.org/10.1109/JPROC.2014.2371999>
- [13] Rahman, A., & Williams, L. (2016). Security practices in continuous deployment pipelines. *Proceedings of the 1st International Workshop on Release Engineering*, 4–7. <https://doi.org/10.1145/2962695.2962707>
- [14] Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>
- [15] Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3), 160. <https://doi.org/10.1007/s42979-021-00592-x>
- [16] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). Machine learning and deep learning approaches for cybersecurity: A review. *IEEE Access*, 10, 19572–19585. <https://doi.org/10.1109/ACCESS.2022.3151248>
- [17] Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., & Leung, V. C. M. (2018). A survey on security threats and defensive techniques of machine learning: A data-driven view. *IEEE Access*, 6, 12103–12117. <https://doi.org/10.1109/ACCESS.2018.2805680>
- [18] Wang, Z. (2018). Deep learning-based intrusion detection with adversaries. *IEEE Access*, 6, 38367–38384. <https://doi.org/10.1109/ACCESS.2018.2854599>
- [19] Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). *Challenges and solutions when adopting DevSecOps: A systematic review*. *ACM Computing Surveys*, 55(3), 1–41. <https://doi.org/10.1145/3496938>
- [20] Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2021). Challenges and solutions when adopting DevSecOps: A systematic review. *arXiv Preprint*. <https://arxiv.org/abs/2103.08266>