iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** VI **Month of publication:** June 2024

DOI: https://doi.org/10.22214/ijraset.2024.63411

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue VI June 2024- Available at www.ijraset.com

Integrating Security into the DevOps Pipeline

Asheesh Kulakarni[1], Ishwari Kothiwale[2], Prof Pavan Mitragotri[3]
[1, 2]*Students,* [3]*Proffessor, Department of MCA, KLS Gogte Institute of Technology, Belagavi 590008*

Abstract: ***A culture known as "DevOps" helps an organization increase the speed at which services are delivered. It promotes collaboration and coordination between the development and operations teams in order to accomplish tasks more quickly and automatically. Despite becoming a fundamental component of all software life cycles, DevOps processes eventually become vulnerable to data security breaches. The integration of security practices into the DevOps pipeline, or DevSecOps, is becoming a vital strategy for guaranteeing the security of software applications in the dynamic and quickly changing technological environment of today. This paper offers a thorough introduction to DevSecOps, describing its guiding principles, methods, and workable implementation techniques. It examines the need for security to be integrated into the DevOps lifecycle and shows how conventional DevOps methods fall short when it comes to resolving security issues. Key DevSecOps principles are covered in the paper, with a focus on cooperation, automation, ongoing monitoring, and cultural shifts toward security consciousness. It explores different approaches to incorporating security into the DevOps pipeline at every step.***
Keywords: Include DevOps, DevSecOps, Security, Continuous Delivery, Continuous Integration

I. INTRODUCTION

DevOps" stands for "development and operations," and it describes a process where software developers and operations work closely together. Improving communication and integration between development and operations is the goal; as a result, all the advantages of contemporary software development methodologies—which involve frequently releasing new software features to end users and then learning from them—can be attained. the great majority of the issues had already been recognized. Prior research has indicated that inadequate cooperation and delayed operations team involvement in the software development process negatively affect software release schedules and quality in the real-world setting. Aiming to close the traditional divide between software development (Dev) and IT operations (Ops), the DevOps philosophy represents a revolutionary change in the software development and operational landscape. The fundamental tenet of DevOps is the smooth integration of these two fields, which promotes teamwork, productivity gains, and quicker deployment times. Continuous integration and continuous deployment (CI/CD), automation, and ongoing monitoring make this integration possible with the goal of delivering high-quality software fast. The DevOps methodology has become essential for organizations seeking to stay competitive and adaptable to market shifts as they endeavor to fulfill the constantly rising demand for cutting-edge software solutions. Unfortunately, security considerations are frequently neglected due to the rapid pace of development and deployment that is inherent in DevOps, which could lead to vulnerabilities. Apart from their expertise in coding, they should also have experience in system administration, infrastructure management, and the use of a variety of DevOps tools. The DevOps engineer is critical to the pipeline's success because they must understand the intricacies of the technology stack, serve as a liaison between the development and operations teams, and adapt to a constantly changing environment. Development (Dev) and Operations (Ops) integration has become a transformative approach in the constantly changing field of software development, giving rise to DevOps. The DevOps pipeline is a powerful, automated framework that automates and streamlines software application development, testing, deployment, and monitoring. It is the central component of this revolution.

II. PROBLEM STATEMENT

Despite the agility and efficiency benefits of DevOps methodologies, the rapid pace of development can introduce security vulnerabilities if security is not considered throughout the entire software development lifecycle (SDLC). Traditional security practices, often siloed and conducted late in the development cycle, are ill-equipped to keep up with the velocity of DevOps. This disconnect between development and security teams leads to vulnerabilities slipping through the cracks, potentially resulting in Costly security breaches: Security vulnerabilities can be exploited by attackers, leading to data breaches, financial losses, and reputational damage. Delayed deployments and rework: Security vulnerabilities discovered late in the development cycle can require significant rework and delays to remediate, hindering release schedules. Inefficient use of resources: Security teams may be forced to perform time-consuming manual security assessments, hindering their ability to focus on more strategic security initiatives. Therefore, there is a critical need for a comprehensive approach that integrates security practices seamlessly into the DevOps pipeline.

©IJRASET: All Rights are Reserved

III.BACKGROUND

In today's software development process, there is a growing trend towards frequent and rapid software releases to gather quick feedback from users. Organizations are driven to embrace DevOps and continuous practices [3] to achieve the ability to deliver software rapidly and automatically as soon as changes are introduced to the mainline. This section will delve into the concept and practices of DevOps as outlined in the literature. We will also review empirical studies that discuss the practices, benefits, and challenges of DevOps to provide context for this paper within the existing body of work. Since DevOps is closely linked to continuous practices, we focused on empirical studies that explicitly address DevOps.

A. The DevOps Main Concept

The DevOps concept aims to address the gaps created by the organizational divide between software development, release, and operations processes. From this perspective, the development and operations teams face multiple conflicting objectives such as 'agility vs. stability' and encounter various obstacles like insufficient information flow and test environments. The scientific definition of DevOps is "a mindset that promotes cross-functional collaboration between teams within a software development organization, especially between development and IT operations, to reduce ambiguity, maintain resilient systems, and expedite change delivery". Collaboration and communication between operations and development form the core principles of DevOps.

B. DevOps Practices

Both technical and non-technical processes are referred to as "DevOps". The categories provide a distinction between procedures that development and operations teams carry out simultaneously and procedures that are unique to their respective domains. The group of common activities is further separated into automated procedural practices and collaborative practices requiring human interactions. More focus is paid to the procedural category's common practices than to its distinctive ones [24].

DevOps methodologies are integrated into the deployment pipeline because they emphasize not just the environments and tools that facilitate software development, deployment, and operation, but also the design and implementation of system features. The software update process in its entirety, from version control to end-user visibility, is represented automatically by the deployment pipeline.

In order to prevent (or reduce) manual system handovers from development to operations, DevOps techniques involve deployment pipeline automation, such as automatic environment provisioning. In order to enable the automatic deployment of changes to virtual machines (VMs) in production or other cloud environments in cloud-based systems, the automated deployment mechanism is integrated into the continuous integration (CI) server using pre-defined triggers.

C. Agile approach

When software inventors saw the failings of cum bersome procedures like the Waterfall fashion, they developed Agile. ways that made software revision far more doable and quick were needed. People are valued further than procedures and outfits, as well as immediateness and inflexibility in responding to and indeed drinking changes rather than clinging to a set course of action. nimble development rejects the notion of a " finished product " and advocates for the incremental and iterative creation of software.

Because nimble software development brigades calculate on continuous delivery, integration, test robotization, and automated builds, they round the ideas of DevOps.

The void left by the demand for further communication between IT operations and development is filled by DevOps. To ice safe and dependable environments for testing and other brigades like marketing and client support as they roll out new software, DevOps regularly communicates with IT operations.

D. Continuous Integration (CI)

"Continuous Integration" (CI) is a software development where platoon members integrate their work on a regular base, generally formerly a day or further, leading to numerous integrations every day. To find integration crimes as soon as doable, an automated figure validates every integration. It aims to incorporate source law constantly into the main branch, allowing the inventor to commit the law several times a day, followed by an automated figure and testing process, and furnishing timely feedback to the inventor if a bug is discovered. The married law is made available for use in products if no bugs are set up. Integration must be ongoing for nonstop Delivery to be.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

E. Continuous Delivery or Continuous Deployment (CD)

If enough development and deployment structure is in place, " Continuous deployment "(CD) refers to the capacity to release a point as soon as it's ready. It's an operations procedure whereby release campaigners who have experienced nonstop delivery reviews are routinely and instantly stationed in a product terrain. The specifics of this deployment may differ based on the technological configuration.

Generally speaking, this means making it astronomically accessible to guests, but this isn't always the case, and in numerous situations, the idea itself isn't indeed applicable.

When the law clears the testing stage and is pre-pruned for deployment into the target surroundings, it can be transferred continuously from an interpretation control system to the product via the figure system using a suite of tools called the nonstop Deployment Pipeline. Continuous deployment, which generally involves delivering the new figure into the product following a successful nonstop integration pass can also involve automated procedures including installing and configuring operating systems, drag porting software, and libraries, as well as virtual machine deployment.

F. Common challenges in DevOps

DevOps, known for its speed and agility, can introduce security vulnerabilities if not addressed properly. Here's how security becomes a challenge in DevOps:

1) *Security vs Speed:* The pressure to release features fast can lead to security being an afterthought. Security testing might be seen as a bottleneck, slowing down deployments. DevOps Silo: Traditionally, security is a separate stage, often happening late in development. This means vulnerabilities might be found late, increasing fixing costs. DevSecOps aims to integrate security throughout the process.
2) *Lax Access Controls:* DevOps environments often grant broad access for efficiency. This can be risky if proper controls and privileged access management aren't in place. Leaks or compromised accounts can be disastrous.
3) *Cloud Security Concerns:* Cloud adoption in DevOps brings its own security challenges. Shared responsibility models and securing cloud resources require extra focus. Misconfigured cloud resources can be a major security risk.
4) *Tool Vulnerabilities:* The vast array of DevOps tools can have vulnerabilities themselves. These tools need security evaluation and updates to stay patched.
5) *Secrets Management:* DevOps workflows involve sensitive information like passwords and API keys. Poor practices, like storing them directly in code, can expose the system to compromise.

By addressing these challenges and adopting a DevSecOps approach, where security is integrated throughout the development lifecycle, organizations can achieve the benefits of DevOps while mitigating security risks. This involves cultural shifts, implementing security best practices, and using the right tools for the job.

G. Challenges in traditional DevOps regarding security vulnerabilities

Although traditional DevOps emphasizes efficiency and speed, it can also introduce security weaknesses. Here are some significant obstacles in this field:

1) *Security vs. Speed:* The focus on quick development and implementation may cause security to take a backseat. One may argue that security testing is a bottleneck that slows down releases.
2) *Absence of DevSecOps Integration:* Security has historically been handled separately after the development life-cycle. This implies that vulnerabilities can be undetected until much later, at which point fixing them would be costly and time-consuming. Security ("Sec") must be integrated throughout the DevOps process (DevSecOps).
3) *Weak Access Controls:* Teams working in development and operations frequently need to have extensive access in DevOps settings. If appropriate access restrictions and privileged access management aren't put in place, this might get dangerous.
4) *Cloud Security Considerations:* Using the cloud in DevOps poses unique security risks. More focus is needed on shared responsibility models and cloud resource security. Misconfigured cloud resources may pose a serious threat to security.
5) *Security of DevOps technologies:* The wide range of DevOps technologies themselves can create vulnerabilities. It is necessary to assess these tools for security vulnerabilities and keep them updated with security fixes.
6) *Secrets Management:* Handling private data, such as API keys and credentials, is a part of DevOps processes. Inadequate techniques for managing secrets, such as hardcoding them into code, might leave the system vulnerable to attack

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IV. SECURITY

Security is seen as a pivotal element of software quality, and in certain situations, it may indeed be the most significant non-functional demand of the system. It is necessary to calculate the functionality of software systems in critical surroundings. Should any security vulnerabilities live, their actuality could heighten the liability of implicit detriment.

Software non-functional conditions, similar to security requirements, can not be introduced into the system in the same way as functional conditions. Because of this, the development platoon should concentrate more on security issues at every position of the software development life cycle. There are several factors. Large-scale, secure operation development requires a secure platform, which serves as a base. This platform consists of both software and tackle.

A. Security In DevOps

DevOps surroundings, while promoting dexterity, can struggle to integrate security effectively. A major chain is neglecting security during the pivotal planning and design phases. This reactive approach leads to slower delivery or releases with security vulnerabilities. Traditional security styles frequently disaccord with DevOps ' presto pace. also, artistic resistance due to delivery pressure and lack of clear security power can hamper visionary security measures. ignorance of security practices among DevOps platoon members can further elevate pitfalls. These factors combine to produce a situation where security sins might remain undetected for extended ages, potentially leading to data breaches and nonsupervisory non-compliance.

B. DevOps Security Threats

1) Insecure Code

a) Malicious Code Injections: Exploiting application weaknesses, attackers insert harmful code to gain control or steal sensitive data. Classic examples include:

b) SQL Infiltration Attacks: Malicious SQL code injected through user input manipulates databases (e.g., data theft or record alteration).

c) Cross-Site Scripting (XSS) Attacks: Malicious scripts are inserted into web pages, enabling attackers to steal user sessions, redirect users to phishing sites, or deface web pages.

2) Cloud Misconfigurations and Shared Security

a) Configuration Errors: Cloud deployments rely on intricate configurations that manage security controls like firewalls, access controls, and encryption. A single misconfiguration can expose cloud resources publicly. For instance, accidentally making an S3 bucket public can expose sensitive data stored within.

b) Shared Security Model: While cloud providers secure the underlying infrastructure, the responsibility falls on the user to secure their specific workloads within the cloud environment. This necessitates organizations implementing additional security measures to protect their cloud-based data and applications.

3) DevOps Toolchain Security Gaps

a) Insecure Secrets Management: Secrets like API keys, passwords, and other critical credentials are essential for DevOps tools to function. Storing these secrets directly in code, configuration files, or scripts makes them vulnerable if an attacker gains access. Secure practices involve using dedicated secrets management tools that encrypt and control access to these credentials.

b) Lax Access Controls: DevOps environments often involve multiple tools and developers with varying access levels. Overly permissive access controls, where everyone has the same level of access, can create security gaps. For instance, a developer with read-only access might unintentionally (or maliciously) grant themselves higher privileges if access controls are not properly configured.

c) Container Vulnerabilities: Containers are a popular technology for deploying applications in a lightweight and portable manner. However, containers can inherit vulnerabilities from their base images, the underlying operating system they are built on. Additionally, misconfigurations in container security settings can also expose vulnerabilities.

4) Additional Threats.

a) Insider Threats: Malicious actors within an organization with legitimate access to systems and tools can pose a significant security risk. These individuals can leverage their access to steal data, disrupt operations, or deploy malware.

b) Denial-of-Service (DoS) Attacks: These attacks overwhelm applications with traffic, making them unavailable to legitimate users. DoS attacks can target DevOps tools and infrastructure, disrupting the software development and deployment process.

C. Integration Of Security

To support the protection of operations and their data during all phases of operation, associations aiming to achieve a secure DevOps terrain integrate tools and procedures. Planning and design are the first stages of integration, which continues throughout an operation's continuance and includes nonstop integration and delivery. Development and operations procedures are combined with security to produce the strategy depicted in the following figure. Integrated security in DevOps primarily focuses on moving it from the end of a software delivery channel to the morning. All stages — planning, development, testing, deployment, and operation — include security. Whenever possible, security should be automated with the aid of testing and covering tools. Defended DevOps demands lesser translucency and more transparent communication toward a common thing. All platoon members involved in the development and delivery of operations should view security as a participated responsibility. Integrating security procedures into the software delivery channel for developed operations With the use of testing and monitoring tools that offer perceptivity into vulnerabilities, DevOps enhances beforehand and methodical threat operation. It might also encourage the development platoon to borrow a security-conscious mindset by encouraging them to consider security issues constantly. Accordingly, during stages like planning, development, and testing, platoon members like developers and testers are more likely to be apprehensive of security pitfalls.

D. Frameworks Used For DevOps Security

Traditional DevOps methodologies, while promoting speed, can introduce security vulnerabilities. To address this, several frameworks advocate for integrating security throughout the development lifecycle. Here are some prominent frameworks for DevSecOps:

1) *NIST Cybersecurity Framework (CSF):* This non-prescriptive framework from the National Institute of Standards and Technology (NIST) provides a high-level structure for managing cybersecurity risk. It outlines five core functions: Identify, Protect, Detect, Respond, and Recover. Organizations can adapt the NIST CSF to their specific DevOps environment.
2) *CIS Controls:* The center for Internet Security (CIS) offers a prioritized set of configuration recommendations for various IT systems and applications. These controls can be integrated into DevOps pipelines to ensure security best practices are followed throughout the development process.
3) *SLSA (Supply chain Levels for Software Artifacts):* This framework from Google focuses on securing the software supply chain. It defines various levels of security practices that can be implemented at different stages of software development, from code signing to provenance verification. SLSA helps ensure the integrity of software artifacts throughout the DevOps pipeline.
4) *Open Web Application Security Project (OWASP) Top 10:* This widely recognized industry standard lists the ten most critical web application security risks. By integrating security testing tools aligned with the OWASP Top 10 into the DevOps pipeline, developers can identify and address these vulnerabilities early in the development process.
5) *Threat Modeling:* This proactive approach involves identifying potential threats and vulnerabilities in a system during the design phase. By incorporating threat modeling into DevOps practices, security considerations are addressed from the very beginning of the development lifecycle.

These frameworks provide different approaches to integrating security into DevOps. The best choice for your organization will depend on your specific needs and priorities. Remember, DevSecOps is not a one-size-fits-all solution, but rather a cultural shift towards collaboration and shared responsibility for security throughout the software development lifecycle.

E. Benefits and Challenges Of DevOps Implementation

1) *Benefits*

a) *Early Detection and Mitigation of Security Issues:* Early Discovery and Mitigation of Security Issues Organizations may identify and address security vulnerabilities and enterprises beforehand in the software development lifecycle because DevSecOps integrates security principles into every step of the process. By being visionary, you can lessen the chance of security breaches and the effect of possible security events.
b) *Improved Security Posture:* Through the integration of security into DevOps processes, enterprises may ameliorate their entire security posture. Organizations may enhance their security defenses by relating and addressing security pitfalls more efficiently through the perpetration of nonstop security testing, robotization of security procedures, and collaboration across development, operations, and security brigades.
c) *Faster Time to Market:* promotes automation, collaboration, and nonstop delivery, enabling organizations to accelerate the delivery of software operations to request. By integrating security into the CI/ CD channel, associations can ensure that security conditions are met without immolating speed or dexterity, performing in faster time to request for new features and updates.

d) *Cost Savings:* Proactively addressing security issues before hand in the development process reduces the st of remediating security vulnerabilities latterly in the software development lifecycle. By automating security testing, associations can identify and remediate security vulnerabilities more efficiently, minimizing the cost and impact of security incidents and breaches.

e) *Enhanced Compliance and Governance:* By integrating security and compliance checks into the development process, DevSecOps styles help enterprises maintain compliance with norms and nonsupervisory scores. The peril of non -non-compliance and forfeitures is dropped by automated compliance testing, inspection trails, and attestation that guarantees security programs are applied constantly.

f) *Cultural Transformation:* The culture of participated accountability and cooperation between the development, operations, and security brigades is fostered by DevSecOps. Associations may apply a security-apprehensive and regard suitable culture in which everyone takes responsibility for security by dismembering organizational silos and encouraging cross-functional collaboration

2) *Challenges*

a) *Complexity and Integration:* Integrating security practices into DevOps workflows requires associations to navigate complications related to tool integration, process alignment, and artistic change. enforcing DevSecOps effectively frequently involves prostrating specialized, organizational, and artistic challenges associated with integrating security into DevOps channels.

b) *Skills and Expertise Gap:* DevSecOps may bear associations to acquire new chops and moxie in security robotization, trouble modeling, secure rendering practices, and security testing. Bridging the chops gap and up-skilling platoon members to effectively apply DevSecOps practices can be a significant challenge for associations, particularly those lacking security moxie.

c) *Tooling and Technology Stack:* opting the right tools and technologies for enforcing DevSecOps can be grueling, given the wide range of security tools and solutions available in the request. Organizations must estimate their conditions, consider factors similar to scalability, compatibility, and integration capabilities, and choose tools that align with their DevSecOps objects and structure.

d) *Compliance and Regulatory Constraints:* Constraints Organizations operating in regulated diligence face fresh challenges related to compliance and nonsupervisory constraints when enforcing DevSecOps. icing that DevSecOps practices misbehave with nonsupervisory conditions and assiduity norms while maintaining dexterity and speed can be a delicate balancing act for associations subject to strict compliance authorizations.

e) *Resistance to Change:* Resistance to change from stakeholders, platoon members, and organizational leadership can hamper the relinquishment of DevSecOps practices. Overcoming resistance to change requires effective communication, education, and leadership buy-in to ensure that all stakeholders understand the benefits of DevSecOps and are committed to its perpetuation.

V. CONCLUSION

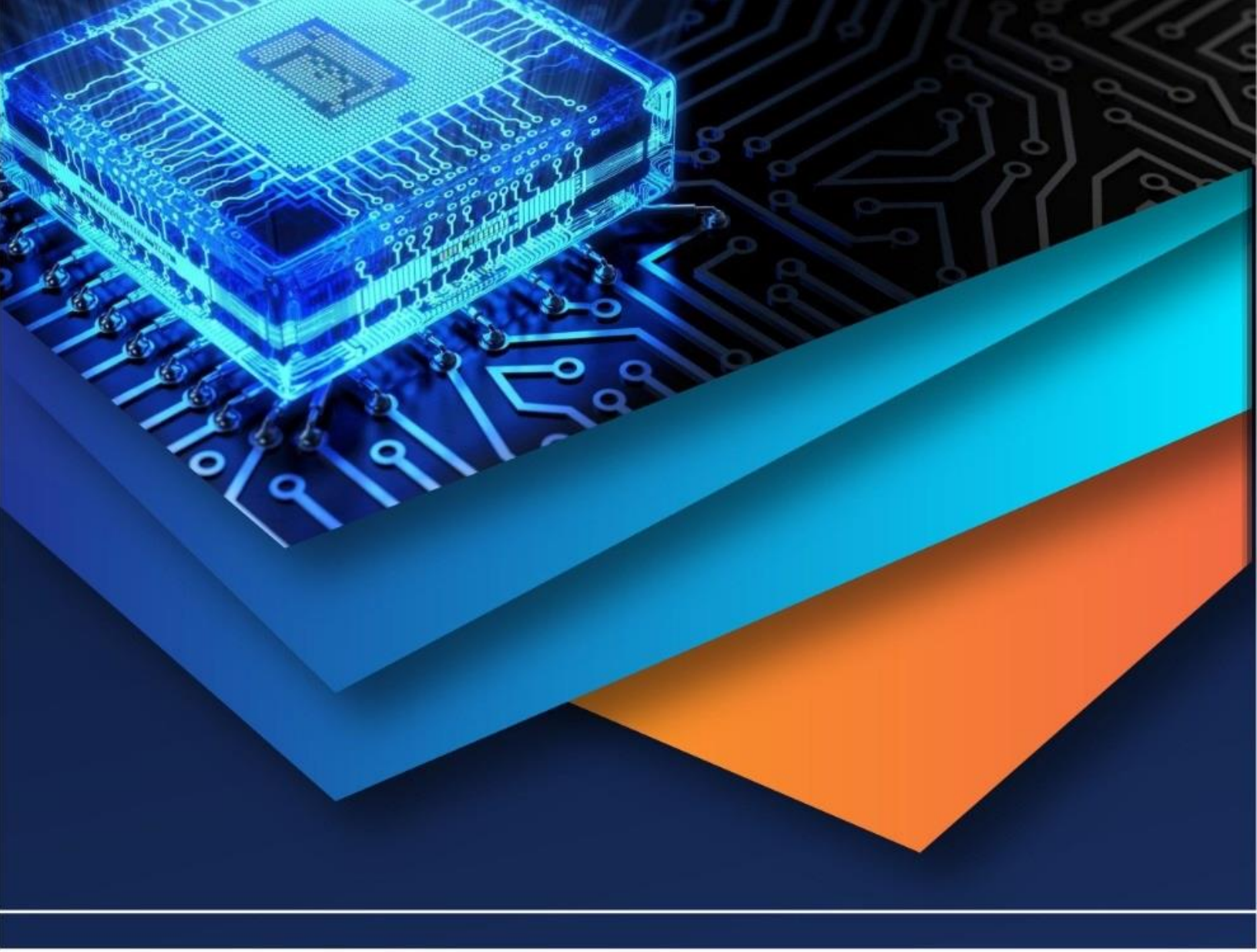
In software development, integrating security into the DevOps channel through DevSecOps has surfaced as a transformative approach, addressing the long-standing challenge of balancing rapid-fire invention with robust security. This paper has explored the principles and practices of DevSecOps, emphasizing its vital part in fostering a more secure, effective, and cooperative terrain for software development. Through espousing practices similar to Security as Code, robotization of security testing, and the early integration of security measures, DevSecOps offers a promising result to the complications of ultramodern software development, icing that security considerations evolve in tandem with technological advancements.

The trip towards completely integrating DevSecOps presents its unique challenges, including integrating appropriate tools, bridging skill gaps, and prostrating artistic resistance within associations. still, associations can overcome these hurdles by espousing a strategic approach that includes fostering a culture of security mindfulness, electing scalable and compatible security tools, and embracing an incremental perpetration process. This strategic approach enhances the security posture of software products. It contributes to the overall speed and effectiveness of the development process, demonstrating the value of bedding security within the DevOps channel. In conclusion, the relinquishment of DevSecOps marks a significant shift towards a more secure and flexible approach to software development. By bedding security practices at every stage of the software development lifecycle, organizations can alleviate pitfalls, enhance compliance, and foster a culture of collaboration and participated, and responsibility among all stakeholders. As the digital geography continues to evolve, the principles of DevSecOps give a roadmap for organizations seeking to navigate the complications of software development, ensuring that they can meet the demands of a decreasingly digital world while upholding the loftiest norms of security and effectiveness

REFERENCES

[1] Z. Ahmed and S. C. Francis, Integrating Security with DevSecOps: Techniques and Challenges, Proceeding 2019 Int. Conf. Digit. Landscaping Artif. Intell. ICD 2019, pp.178 - 182, 2019.
[2] Ashfaque, A., Rahman, U., Williams, L: Integrating Security with DevSecOps: Techniques and Challenges Software security in DevOps: synthesizing practitioners' perceptions and practices. , Proceedings of the International Workshop on Continuous Software Evolution and Delivery, CSED 2016, pp.70–76. ACM, New York (2016).
[3] 3. Cham Rajapakse, R. N., Zahedi, M., Babar, M. A., Shen, H. (2022)., Challenges and solutions when adopting DevSecOps: A systematic review. Information and Software Technology, 141, 106700.
[4] H. Myrbakken and R. Colomo-Palacios, DevSecOps: A multivocal literature review, P09 2017, pp. 17–29.
[5] H. Yasar and K. Kontostathis, Where to integrate security practices on DevOps platform," International Journal of Secure Software Engineering, vol. 7, pp. 39–50, 10 2016.
[6] Hajjdiab and Al Shaima Taleb, "Adopting Agile Software Development: Issues and Challenges", International Journal of Managing Value and Supply Chains, , vol. 2, no. 3, pp. 1-10, 2011.
[7] U. Rahman and L. Williams, "Software security in DevOps", Proceedings of the International Workshop on Continuous Software Evolution and Delivery, 2016.
[8] L. Zhu, L. Bass and G. Champlin-Scharff, "DevOps and Its Practices", IEEE Software, , vol. 33, no. 3, pp. 32-34, 2016.
[9] M. Airaj, "Enable cloud DevOps approach for industry and higher education", Concurrency and Computation: Practice and Experience, , vol. 29, no. 5, p. e3937, 2016. 32-34, 2016.
[10] S. Gutta, S. Prasad and J. Angara, " DevOps product line engineering (DPLE): Where devops meets software product lines", PONTE Inter national Scientific Research Journal, 72, no. 11, 2016.
[11] G. Linkevics, "Adopting to Agile Software Development", Applied Computer Systems, vol. 16, no. 1, pp. 64-70, 2014.
[12] J. Humble, J. Molesky, Why enterprises must adopt DevOps to enable continuous delivery, Cutt. IT J. 24 (8) (2011) 6–12.
[13] P. Debois, Agile infrastructure and operations: how fragile are you? in: Agile 2008 Conference, IEEE, 2008, pp. 202–207, doi:10.1109/Agile.2008.42.
[14] D.E. Strode, S.L. Huff, B. Hope, S. Link, Coordination in co-located agile software development projects, J. Syst. Softw. ,85 (6) (2012) 1222–1238.
[15] L. Bass, I. Weber, L. Zhu, DevOps: A Software Architect's Perspective, Addison-Wesley Professional, 2015.
[16] X. Larrucea, A. Berreteaga, and I. Santamaria, "Dealing with Security in a Real DevOps Environment", EuroSPI 2019: Systems, Software and Services Process Improvement, Springer, vol: 1060, pp: 453–464.
[17] Yarlagadda, R. Teja, "DevOps for Better Software Security in the Cloud", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.7, Issue 9, page no.1081- 1085, September 2020, Available at SSRN: https://ssrn.com/abstract=3807615.
[18] R. N. Rajapakse, M. Zahedi, M. A. Babar, "An Empirical Analysis of Practitioners' Perspectives on Security Tool Integration into DevOps", Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), October 2021, Article No: 6, pp: 1–12, https://doi.org/10.1145/3475716.3475776.
[19] Mohan, V., Othmane, L. B. (2016). SecDevOps: is it a marketing buzzword? -mapping research on security in DevOps. In Availability, Reliability and Security (ARES),vol. 29, no. 5, p. e3937, 2016.
[20] M. Airaj, "Enable cloud DevOps approach for industry and higher education", Concurrency and Computation: Practice and Experience, 2016 11th International Conference on (pp. 542-547).
[21] H. Yasar and K. Kontostathis, "Where to integrate security practices on DevOps platform," International Journal of Secure Software Engineering,vol. 7,pp.39–50,10,2016

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)