IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

Devsecops For Secure Application Delivery

[1]Vismaya Raj, [2]K Kanagalaksmi
[1]MCA Student, [2] Professor
School of Science and Computer Studies
CMR University Bengaluru

Abstract: DevSecOps is a paradigm shift in software development where security is now built in security into every part of the software development lifecycle (SDLC). This approach has now superseded the old ineffective, counterproductive systems of security and has incorporated a preventative, shift-left approach in which security becomes a collaborative activity between the development, security, and operation teams. DevSecOps supports application security in such a way that automated security scanning and continuous monitoring have been built in and by doing so, it minimizes the number of vulnerabilities, increases security, and shortens the deployment cycle. This white paper expounds on the underlying principles, advantages, obstacles, and the key practices of DevSecOps leading to an in-depth case study of applying the same to a contemporary e-commerce application, providing an exhaustive blueprint that an organization would follow in its bid to deliver secure high-quality software in a super-fast way.

Index Terms: **DevSecOps, Application Security, CI/CD, Container Security, Docker, Security Automation**

I. Introduction

DevSecOps is a convergent software development model that inculcates the concept of security in DevOps. Development, Security, and Operations also known as DevSecOps is a collaborative system where security becomes a central part, as it is no longer just an afterthought, a central part at all tiers of the software development process.

The presentation of DevSecOps can only be characterized as an answer to the pressure that comes with certain standard DevOps implementations. However, DevOps is revolutionary in terms of its approach to software-delivery by focusing on speed, efficiency and alignment between development and operations; the resultant speed has, however, incidentally made security a second tier process or led to bottlenecks. The accelerated release schedules and the microscopic complexity of cloud-native environments only compounded that problem, and screening security becomes hard to do at the end of the process. DevSecOps tackles this directly by incorporating the pace and rapidity of DevOps with an extra security feature, whereby the applications are deployed rapidly but they are protective by nature. This kind of evolution knows that a fast delivery will be needed but cannot be sustained and will always be expensive unless there will be good security systems in place. Companies that implement DevOps without introducing the element of security at the early stages face the risk of high technical debt and devastating security breaches that negate the speed advantage that they were even working toward in the first place.

The key principles behind DevSecOps are cross-functional teamwork, continuous security and automation of security. Automation gives reliable and repeatable security practice, like automated configuration management, vulnerability scan, and testing. Continuous security means assuming security must be taken into consideration continually through the whole lifecycle, rather than as a point in time test. Lastly, cross functional teamwork enables the culture of shared ownership of security without the walls between the development and operations and the security teams.

II. Literature Review

This situation has seen the emergence of the DevSecOps because of the failure of the traditional methods of security. According to one study by Microsoft (2022) [1], and AWS (2023) [2], the traditional practices of security operations are typically performed on the final stage of software development that contributes to delayed launching to the market and costly fixes taking months in that process. On the other hand, DevSecOps sets a good and collaborative culture, as all the Software Development Lifecycle (SDLC) phases include security [4] (CrowdStrike, 2023).

The problem in the context of the literature is directed to the fact that a need has occurred to substitute reactive security with constant and automatized actions that can be conducted within the framework of CI/CD pipeline. Many of the works also suggest that increased agilityfr, reduced risk and high compliance are the direct positive attributes of this strategy.

III. Challenges

A DevSecOps adoption process is an organizational change that although laden with high costs of potential improvements (is not) without its share of challenges based on the level of organizational culture and competency sets as well as technology challenges.

a). Organizational and Cultural problems

The most likely pitfalls are represented by cultural resistance to change. DevSecOps requires that security is an everybody job, the polar opposite of the walled-garden of yesteryear tradition of security being a set-apart, specialized role. Such an alteration is resisted, and teams regard security integration as a speed break or another overhang.

A strong security as an afterthought culture stands in the way. In the past security was more of an after thought and it takes a lot of effort to shake this already deep rooted culture. This helps in a kind of an us-versus-them culture between security and development teams since they use different roadmaps, have different accountabilities, and are driven by different motivations. Such a conflict may fail effective cooperation, even at enterprises whose DevOps cultures are very developed.

There is also a perception that there is usually an impediment of security on innovation or slows down growth. Security is even sometimes viewed as an inhibitory factor to agility as the need to execute software much faster and frequently occurs. This contributes to living up to a move of integrating security further into the pipeline.

All these cultural issues are highly interdependent and therefore, lead to a vicious cycle. Such an approach of security as an afterthought perpetuates the us-versus-them mentalities since security teams are perceived as late-stage blockers, which causes inner conflict. That pressure will in its turn reinforce the belief that security causes a delay and the cultural change will be the more complex. The absence of executive buy-in then denies any efforts to break such a cycle the resources and strategy synchronicity that it needs. To overcome these cultural difficulties, it is important to have a multiple-angled plan of addressing the organization objectives in expressive way to enhance open communication, continuous training, and active empathy and ownership through a top-down manner. It is a long-term commitment to organization change management.

b). Skills and Knowledges

A major real-life problem is that many developers and operations personnel lack appropriate security training and they do not possess a deep knowledge of security. Such groups though are very well-paced in their area of expertise are not always good at maneuvering their way through high-level security knowledge so that security may be incorporated in their daily operations. Such a talent gap extends to a wide range of functions, including business stakeholders and auditors. Another issue is that organizations constantly lack the right security guidance as they lack the resources, well established standards, and security monitoring to actually practice security in an adequate manner throughout the SDLC.

c). Tooling and Integration complications

The technical environment of DevSecOps is intimidating [9] At this point, we have a problem of sprawl, where large organisations end up with a bunch of siloed tools to support security and DevOps. Fragmentation makes integration difficult, resulting in inefficiency, waste of duplication of effort, and gaps in security blanket coverage.

There is also a potential of frustration in automation because most of the traditional security mechanisms have been developed to be manual and may be a bit hard to automate. This is a conflict between ideal velocity of DevOps and necessary security checks which slow down the development rates when not properly addressed.

The other major hindrance is legacy infrastructure constraints Older systems and monolithic architectures might not have the degree of flexibility or API-style features needed to integrate easily with a modern DevSecOps approach, and as such integration problems can become very complicated and the introduction of automated security can become impossible. Siloed operations and a failure to make strategic plans are some of the causes of these tooling issues. Dev, security, and ops choose tools based on the needs of their own group and without taking into account cross-functionality or taking advantage of an integrated toolchain. The result of such work is a patchwork of tools that actively works against collaboration and automation. The issue here is not what tools to take, but how and who these tools integrate and who they are supported by. Technology Part Two: There is thus the need to have organizations that have an end-to-end tool chain strategy, which prioritizes more on multi-analysis engine supported platforms that would allow transparent integration in CI/CD pipelines. This also has to involve the disappearance of isolated point solutions usage in favor of integrated platforms and the shift towards the model of governance in which the tools will be chosen in accordance with the overall commercial enterprise DevSecOps vision.

d). Getting There Fast vs. Getting There safely

The eternal dilemma between speed and security never goes away in DevSecOps.Rushing teams that have strict timelines to meet in terms of delivery may decide to go with rapid feature release, presumably or inadvertently pushing or moving courteousness or security concerns to the back burner and one of the reasons why vulnerabilities are introduced. It is further evident there is a constantly evolving cybersecurity landscape worsening this. New threats and vectors emerge routinely, and maintaining awareness of them and developing respective security measures to deal with new ones is an ongoing task. It demands attention all the time and proactive stance concerning threat intelligence and security posture modifications.

IV. Methodology

DevSecOps has gone a long way to come up with an effective and successful strategy, which has seen it using a comprehensive approach, incorporating the three fundamental elements of people, process and technology. They are important best practices in implementing them as follows:

a). Shift Security Left: Place more of a focus on consideration and practice of security at the initial stages of the development process. This involves threat modeling in the planning stage, establishing standards of secure coding documentation, and carrying out automated security testing as early as it is achieved. In this active approach, cost and effort to carry out remediation is considerably less.

b). Automate All You Can: Combine and automatically test security (SAST, DAST, SCA) and policy and enforce in the whole delivery pipeline. The key to removing friction and making sure the delivery is fast and consistent in terms of security is automation.

c). Develop a Culture of Shared Responsibility: This requires the incorporation of shared responsibility culture with the break-down of the old organizational silos that existed and the encouragement of collaboration and open communication between the development, security, and operation teams. Security has to be treated as the shared mission, not as the task of one team.

d). Go to the Developers: Offer security knowledge, in-depth remediation advice, and immediate feedback within the current workflow and IDE of the developer. This enables the developers to rectify problems in the most effective way possible and it motivates them to assume ownership of security. It is also important to invest in the relevant security training with the developers.

e). Use Platform-Based Application Security Testing (AST): A centralized SaaS service with back-end multiple engines (SAST, DAST, SCA, IAST) should be adopted over individual, standalone products. These platforms harmonize effectively within CI pipelines, and have policies controlled centrally making the process of managing them easy and risk information as priorities.

f). Install Continuous Monitoring and Incident Response: Create sound mechanisms on continuous monitoring of applications in the production environment in regard to vulnerability and threats. Effective response to the security incidents that includes fast detection, containment, and recovery requires the presence of a conclusive incident response plan.

g). Incorporate Zero Trust Principles: Employ principle of least privilege (PoLP) to all systems and ensure that all users and services acquire the bare minimum of privileges. Take multi-factor authentication (MFA) everywhere and think about reorganizing the networks using Zero Trust Network Architecture (ZTNA) concepts, such as micro vaulting, where they restrict future movement in the event of a compromise.

Adopt the Use of Infrastructure as Code (IaC) Security: Use the same level of security on IaC that we are accustomed to doing with application code. Use IaC model to scan templates and automatically enforce security policies to ensure consistent infrastructure deployment that is secure.

h). Implement Feedback Loops: Feedback loops should be established to give developers actionable information on security tooling in real-time. Most important security lessons need to be learned during security incidents and monitoring in production and provide input to the development and planning processes that could lead to improving and evolving toward new threats.

V. Implementation

Scenario Overview: Protecting an E-Commerce App

In the interest of demonstrating the practical implementation of DevSecOps one can mention an example of an e-commerce app ShopSecure which was rapidly expanding in size and was experiencing a problem of discovering the vulnerabilities late in the software pipeline and experiencing a slowness in relation to compliance. With the aim of addressing these issues and in-turn ensure that its secure features are efficiently distributed, ShopSecure replaced its legacy security model into a full DevSecOps pipeline [10]. The architecture guided by the micro services was containerized with the docker, was orchestrated using Kubernetes, and was deployed on the such cloud platforms like AWS or Azure with the help of CI/CD pipeline.

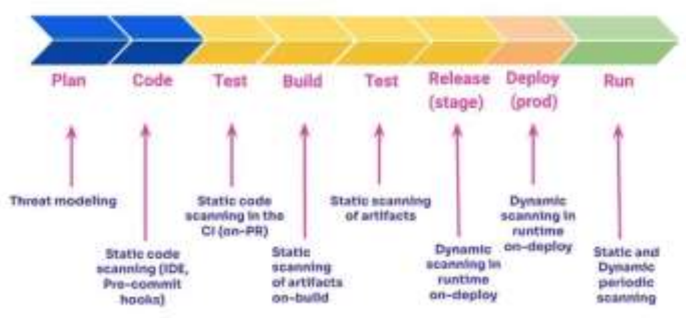
Plan
Code
Test
Build
Test
Release (stage)
Deploy (prod)
Run
Threat modeling
Static code scanning (IDE, Pre-commit hooks)
Static code scanning in the CI (on-PR)
Static scanning of artifacts on-build
Static scanning of artifacts
Dynamic scanning in runtime on-deploy
Dynamic scanning in runtime on-deploy
Static and Dynamic periodic scanning

Fig.1: Phase-Wise Integration of DevSecOps Practices

a). Planning/Designing:

The threat modeling was conducted during their initial stages using some of the frameworks like STRIDE to assist in modeling the potential security concerns of the newly created micro services including the payment gateways. Every user story was associated with security requirements and the mapping of PCI DSS and GDPR compliance carried out during the design. Such tools as OWASP Threat Dragon and Jira provided easy security planning.

b). Age of Development:

Information security measures When it came to securing information, there was no deviating from the practice of secure coding which follows OWASP Top 10. Security plug-ins that issued real-time feedback were incorporated into IDEs developers were using and Static Application Security Testing (SAST) (e.g. SonarQube) tools were used to identify vulnerabilities prior to any code being committed.

c). Security of CI/CD Pipeline:

- Each push to the code would initiate a CI/CD pipeline which would include:
- Third party dependency search Using Software Composition Analysis (SCA)
- Leakage of base image cycles through container image via vulnerabilities scanning
- Security gates that would not allow high risks issues erects
- Actual simulation of attacks by carrying out dynamic application security testing (DAST)
- The pipeline supported such tools as GitLab CI/CD, Snyk, Trivy, and OWASP ZAP.

d). Safe Injection:

Terraform was used to provision infrastructure which was then checked via infrastructure as code templates scanned by tfsec or Checkov. One of the security policy tools was automatic security policy enforcement OPA. HashiCorp vault was used to handle secrets and Role-Based Access Control (RBAC) was used to use close privileges.

e). Secure and monitoring system of operations:

Prometheus, Grafana and ELK stack made monitoring of real-time applications and infrastructure possible. Mechanized disaster response plan to incidents was in existence, and was exercised at regular intervals. In monitoring, there was continuous feedback on the planning process so that the following amendments to the errors made by the incidents could be done easily.

VI. Results:

Implementing DevSecOps at ShopSecure provided the quantifiable benefit of gains in the areas of security, and the pace of development[11], and collaboration among teams. The integration of security as part of the Software Development Life Cycle (SDLC) allowed the company to correct the vulnerabilities prior to production and the cost of correcting its vulnerabilities was much more easy.

Security checks implemented in the CI/CD chain also removed delays that could arise due to security check at the last minute before a final deployment, ensuring a quicker and more consistent release of feature updates. This further helped it reduce time to market directly and continued to comply with high regulation such as PCI DSS and GDPR.

In addition, the move to collaborative culture between development, operation, and security teams led to improved communication, security best practices awareness, and the general increase in proactivity in organizational risk management.

VII. Discussion

The application of DevSecOps in the case study (ShopSecure) therefore leads to realization of significant improvement in the application security levels as well an uplift in the time-to-market aspect. The mentioned theoretical benefits in the literature can be seen in the effort of vulnerability decrease and regulatory conformance and collaboration measurability. Still, the problems like tool fragmentation, resistance to culture and legacy infrastructure remain the issues. Although those concerns are partly covered through automation and training, the priority that businesses should have is to maintain long-term culture change and integration.

Furthermore, the process of DevSecOps is not a one shot thing but a continual journey because it keeps evolving with regard to the evolving threats and upgraded technology. One cannot change through tools alone since the shift in the paradigm is necessary, i.e., to a shared responsibility and security-first mentality.

VIII. Conclusion

The DevSecOps adoption is a paradigm shift in the security of applications that it is not merely a reactive, late-stage afterthought, but rather, an integrated, proactive, continuous operation in the organization. DevSecOps is able to deliver security to software applications by making them secure by design by integrating security throughout the software development lifecycle via the engineering of software applications, including the conception and design stages, as well as coding, construction, testing, and constant upkeep.

An agile development culture that enables development, security, and operations teams to operate with efficacy is the push-left paradigm, implemented through heavy automation, ongoing security frameworks, and a culture of responsible collective work. This synergy can bring greater benefits to business as well as increased security position that is detecting and remediating vulnerability sooner and more effectively. Such gains are reduced time-to-market, significant costs savings due to avoiding expensive violations and rework, better regulatory compliance, and the development of a security-minded organizational culture.

The path to an effective DevSecOps delivery is not without difficulty, especially when it relates to cultural barriers, technical know-how and tooling complexities, but these obstacles can be overcome with a cohesive plan and dedication. The example of the detailed implementation of an e-commerce platform shows the practical application of these principles and tools that can lead to a real result including the acceleration of the deployment, the quality of the code, and the minimization of the security problems.

Finally, the effective DevSecOps plan is not only a technical one but an ongoing trip of enhancing an organization, network, and sharing information. Through these principles, building resilient, high-quality software that delivers results in the modern dynamic world of digital will be possible.

References

[1] xMatters, “The Principles of DevSecOps,” *xMatters*, [Online]. Available: https://www.xmatters.com/blog/the-principles-of-devsecops
[2] Fidelis Security, “DevSecOps in SDLC: Secure Agile Development,” *Fidelis Security*, [Online]. Available: https://fidelissecurity.com/cybersecurity-101/cloud-security/what-is-devsecops/
[3] Amazon Web Services, “What is DevSecOps? - Developer Security Operations Explained,” *AWS*, [Online]. Available: https://aws.amazon.com/what-is/devsecops/
[4] Microsoft, “What Is DevSecOps? Definition and Best Practices,” *Microsoft Security*, [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-devsecops
[5] Black Duck, “What Is DevSecOps and How Does It Work?” *Black Duck Software*, [Online]. Available: https://www.blackduck.com/glossary/what-is-devsecops.html
[6] CrowdStrike, “What is Shift Left? Security, Testing & More Explained,” *CrowdStrike*, [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/shift-left-security/
[7] OpsMx, “What is DevSecOps?” *OpsMx*, [Online]. Available: https://www.opsmx.com/blog/what-is-devsecops/#:~:text=The%20core%20principles%20of%20DevSecOps,vulnerability%20scanning%2C%20and%20configuration%20management
[8] CrowdStrike, “What is Shift Left? Security, Testing & More Explained,” *CrowdStrike*, [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/shift-left-security/#:~:text=Shifting%20left%20in%20the%20context,the%20software%20development%20life%20cycle.
[9] ValueLabs, “Benefits of Adopting DevSecOps For Your Organization,” *ValueLabs*, [Online]. Available: https://www.valuelabs.com/resources/blog/devsecops/benefits-of-adopting-devsecops-for-your-organization/
[10] Veritis, “Securing Energy Services: A DevSecOps Implementation Case Study,” *Veritis*, [Online]. Available: https://www.veritis.com/case-studies/devsecops-implementation-enhancing-security-for-an-energy-services-firm/

[11] Codefresh, “DevSecOps Pipeline: Steps, Challenges, and 5 Critical Best Practices,” *Codefresh*, [Online]. Available: https://codefresh.io/learn/devsecops/devsecops-pipeline/
[12] E. Wags, “Top 10 Challenges of DevSecOps Implementation in 2024,” *DEV Community*, [Online]. Available: https://dev.to/emma_wags_8dd9b74533690da/top-10-challenges-of-devsecops-implementation-in-2024-2h5j