Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2022, 9(6):79-86

Research Article ISSN: 2394 - 658X

Devsecops in the Cloud Integrating Security into Continuous Integration Continuous Deployment (CICD) Pipelines

Rajashekar Reddy Yasani[1], Karthik Venkatesh Ratnam[2]

[1]Sr Cloud Security Engineer
Independent Security Researcher, Dallas, TX
Cloud Security, Cloud Computing, Cyber Security
rajshekaryasani@gmail.com
[2]Cloud Engineer, Independent Security Researcher, Boston, MA
Devsecops (cloud security)
karthikratnam1@gmail.com

ABSTRACT

Development and Operations (DevOps) is a methodology that aims to establish collaboration between programmers and operators to automate the continuous delivery of new software to reduce the development life cycle and produce quality software. Development, Security, and Operations (DevSecOps) is developing the DevOps concept, which integrates security methods into a DevOps process. DevSecOps is a software development process where security is built in to ensure application confidentiality, integrity, and availability. This paper aims to identify and prioritize the challenges associated with implementing the DevSecOps process. We performed a multivocal literature review (MLR) and conducted a questionnaire-based survey to identify challenges associated with DevSecOps-based projects. Moreover, interpretive structure modeling (ISM) was applied to study the relationships among the core categories of the challenges. Finally, we used the fuzzy technique for order preference by similarity to an ideal solution (TOPSIS) to prioritize the identified challenges associated with DevSecOps projects.

Keywords: DevOps, DevSecOps, Challenges

INTRODUCTION

A wide variety of information technology companies are beginning to adopt DevOps in a substantial way. DevOps is being used in some capacity throughout the organizations of early-stage businesses as well as "unicorns" such as Facebook, Amazon, and others. In the case of Flickr, for instance, "the close communication and collaboration between the development and operations teams enhanced the release time of code by a factor of ten [1]." EtsDesy's success story in transitioning to the DevOps culture was described by Rembetsy and McDonnell in a similar manner. It was said in [2] that the majority of companies are eager to begin the DevOps program because of the benefits that are already known about it. Increasing the level of collaboration between the development and operation teams in order to eliminate discrepancies between development, operations, and releases is the primary goal of DevOps, which focuses on the quick creation and delivery of software through the use of agile principles. The following is the definition that we have chosen to use in this article: "DevOps is an effort within an organization that is collaborative and multidisciplinary in nature, with the goal of automating the continuous delivery of new software versions while simultaneously guaranteeing their correctness and reliability." In order to continually produce a software product, take advantage of market opportunities, and reduce the amount of time required to consider customer feedback, the development team, as well as "customers, operations, and quality assurance stakeholders," interact.
In recent years, the DevSecOps paradigm has emerged as a result of the realization that software development teams have grasped the significance of addressing security risks at an earlier stage in the development cycle. The DevSecOps methodology incorporates security management at every stage of the development process in order to

facilitate coordination of activities among the three teams responsible for development, operations, and security [4]. The goal of DevSecOps is to guarantee that software programs are secure before they are provided to the user and that they remain secure throughout the process of application updates. We decided to use the following definition for the purpose of this paper: "DevSecOps is a model on integrating the software development and operational process while taking into consideration security activities: requirements, design, coding, testing, delivery, deployment, and incident response." DevOps processes that have reached maturity include continuously testing, deploying, and validating software to ensure that it satisfies all requirements and enables a speedy recovery in the event of a problem occurred. As a result, we are able to confidently assert that "DevSecOps is DevOps done right" [5].

Although DevSecOps is becoming increasingly important in the software industry, there has been relatively little research conducted to better understand the problems that practitioners confront. To add insult to injury, there has been no empirical research carried out to rank the highlighted challenges in order of importance and investigate the interrelationships between them. With the help of this article, we intend to identify and rank the difficulties that are linked with the implementation of DevSecOps. In this study, our objectives are as follows: (1) to investigate the DevSecOps challenges that have been reported in the multivocal literature and in industry practices; (2) to investigate the relationship relationships between the core categories of the identified challenges; and (3) to rank the identified challenges according to the importance they hold for DevSecOps. It is easier for industry practitioners and academic researchers to concentrate on the most important aspects of the DevSecOps process when the problematic elements are prioritized after they have been identified. Experts in the business are able to build innovative and efficient strategies for the successful execution of DevSecOps-based projects when they have a thorough understanding of the problematic issues associated with DevSecOps.

BACKGROUND

Developers and operations teams work together more effectively via DevOps, which aims to streamline software deployment and updates with little impact on users' experience [6]. The DevOps paradigm requires separate but complementary processes, tools, and knowledge bases for the development and operations teams. Operational teams use the most recent version to preserve project quality and other non-functional criteria, while the development team regularly adds new features into production using DevOps [7]. To put it simply, the goal of development operations is to provide a process where stakeholders in both development and operations can work together to release new features to production.

Integrating security at every stage of the DevOps life cycle—from design to integration, testing, deployment, and delivery—is automated by DevSecOps. Incorporating security measures into DevOps processes is not without its difficulties, though. As an example of a difficulty, getting security to adopt DevOps practices requires very nimble security methodologies and clear, mutually understood processes among the security, development, and operations groups. The next step in integrating security into the company culture is figuring out how the company will begin using DevSecOps methods, which include quickly adapting to new skills, tools, standards, and processes [8]. Further, DevSecOps initiatives constantly face the problem of automating and staying up-to-date with the latest technologies and tools. Due to the ever-changing nature of the environment, it is imperative that security features are prepared and available in tools that are compatible with the appropriate platforms.

For better security team integration within DevOps, McKay [9] offered the following suggestions: (1) Identifying the challenge should be the first step. (2) Early exposure to DevOps should be a priority for security teams. (3) Adopting a centralized solution should be the second step. (4) Instead of reviewing new environments, the security team should transform their security practices into templates. (5) As many software defects as possible should be spotted using software breakdown attempts. Just like this, Goldschmidt and McKinnon [10,11] suggested the following three levels to be considered while building a DevSecOps environment for a cloud infrastructure: (1) At the infrastructure layer, security experts should be involved in infrastructure configuration and deployment to lessen potential risks; (2) At the platform layer, adjustments should be made to settings that include automation and control rather than the platform itself; and (3) At the application layer, specific processes should be prioritized to enhance application security [12,13]. Collaborating across teams, comprehending log files, assessing security capabilities before enabling communication among many systems, configuration management, continuous integration, and monitoring output in a production system are all considered such activities [14].

There has been scant research on the elements that can compromise the DevSecOps processes, which is a shame given the importance of security integration into the DevOps paradigm [15]. We created a taxonomy of DevSecOps difficulties since it is becoming more important to understand and respond to security concerns throughout DevOps efforts. Difficulties discovered by a multivocal literature research and questionnaire survey with industry professionals will form the basis of the taxonomy [16–18]. Finding out how professionals in the field feel about the DevSecOps paradigm is the main goal of the survey. Because people (the DevSecOps team) are better at verbalizing their emotions, quantitative prediction is difficult for them. Because expert opinions are often ill-defined and fraught with uncertainty, rating difficult criteria based on them is no easy task [19]. Hence, to convert the qualitative predictions made by DevSecOps professionals into numerical prioritizing values, this study employs the fuzzy TOPSIS analysis. It is a popular method for assessing decision-making issues with multiple criteria [20].

UNDERSTANDING DEVSECOPS AND ITS ROLE IN CI/CD

Introduction to DevSecOps in CI/CD

The theory or philosophy behind integrating security procedures into the DevOps process is known as DevSecOps. Software development life cycles (SDLCs) that prioritize security and aim for continuous delivery are also described using this term. Security in DevOps is frequently overlooked and regarded as an afterthought. Software development life cycles (SDLCs) typically conclude with information security (InfoSec). Once the software development life cycle (SDLC) is complete, finding security flaws can be a major source of frustration. The goal of DevSecOps is to make security an integral aspect of the SDLC from the very beginning. Processes like CI/CD, or Continuous Integration and Continuous Delivery, have been developed by General DevOps. Agile development relies on continuous testing and verification of code correctness through Continuous Integration and Continuous Delivery.

Why DevSecOps?

It is essential to implement security measures during the earliest phases of the software development life cycle (SDLC) because, in a nutshell, our lifestyles that are dependent on technology would be put in jeopardy if we do not implement them. One of the most significant dangers that governments and businesses are confronted with in the modern era is the occurrence of security breaches. Over the past few years, a number of significant firms have experienced security breaches, which has led to a continued erosion of trust among customers, which in turn has resulted in enormous financial losses occurring annually. It is possible that your product will be vulnerable at the very last minute before DevSecOps is implemented, which could result in several expensive iterations. Following the implementation of DevSecOps, your product will be baked with the highest possible security standards. However, the likelihood of discovering unforeseen problems in the final minutes is significantly lower than it was previously thought. Taking use of DevSecops will, in general, strengthen your credibility in the market and foster confidence among customers. It is a fantastic approach to transition into discussing how DevSecOps fits into the continuous paradigm, so keep all of these concepts in mind.

What is the DevSecOps Pipeline?

Phases such as "Plan," "Code," "Build," "Test," "Release," and "Deploy" were common in a DevOps pipeline. With DevSecOps, you can be sure that every step of the DevOps pipeline is secure. The security checks implemented into the CI/CD pipeline as a result of implementing DevSecOps are explained here.

Plan: During the planning phase, make sure to run security analyses and come up with a strategy to figure out potential testing situations.

Code: Implement and make advantage of linting tools and Git controls to safeguard API keys and passwords.

Build Before releasing code to production, it is subjected to Static Application Testing (SAST) to identify and fix any errors. Programming languages are the exclusive domain of these tools.

Test: In order to find bugs in your application that are connected to authentication, authorization, SQL injection, and API endpoints, you can utilize the dynamic application security testing (DAST) tools.

Release: Scanning for vulnerabilities and conducting penetration tests are both made possible by the security analysis tools. Prior to publishing the program, make sure you use these tools.

Deploy: Send a secure infrastructure or build to production for final deployment after the above runtime tests are completed.

Relationship of DevSecOps with CI/CD Pipeline?

Open source software (OSS) libraries, which we use both for importing and writing, may have security flaws. Manual code reviews aren't scalable, and there are a lot of developers that code every day. This is where DevSecOps truly shines. The security of our software deliveries is enhanced via DevSecOps. Implementing the continuous everything paradigm, continuous delivery pipelines (or CI/CD pipelines) aid in validating every commitment our team makes. Incorporate automated security checks into our continuous pipelines to keep an eye out for potential security holes and provide early warnings when they occur. As your company grows, so can your continuous security approach.

How to Implementing Continuous Security?

It is recommended to begin implementing continuous security in security unit testing. Just like any other unit test we build, the Security unit test has essential requirements.

SAST

Our code and any libraries you import are scanned by the SAST code analyzers for security vulnerabilities. A variety of cutting-edge tools are seamlessly integrated with the continuous delivery pipeline to do this, which is known as SAST (Static Analysis Security Testing). Because these tools are language-specific, pick an SAST scanner that works with the language you intend to use.

A word of caution: Additionally, SAST has the ability to record false positives, which gives pipelines the ability to "remember." It is risky for teams to become so annoyed by false positives that they cease responding to broken pipeline notifications. You should modify the pipeline so that it flags the erroneous notice many times once the team has found it and provided adequate justification.

DAST

In contrast to static analysis security (SAST), dynamic application security testing (DAST) simulates an attacker's actions to audit your application while it is executing. Since they communicate with the external application, DAST (Dynamic Application Security Testing) scanners do not rely on particular languages. Make sure you receive early feedback on any security issues by integrating both ways in our workflow.

DevSecOps is the Future of Security

Everyone is responsible for security in today's environment. A self-proclaimed expert's mindset should not restrict your thinking. After suffering severe repercussions, many active organizations are re-evaluating their security strategy and allocating more resources to it. Security is now a top concern for more than just the company. Integrating it with the continuous delivery pipeline is a must-have for many reasons.

HOW A DevSecOps PIPELINE ADDRESSES THE NEED FOR INTEGRATING SECOPS INTO CI/CD PIPELINES

This blog post will center on the need of incorporating security into software delivery pipelines and how a DevSecOps Pipeline can be engineered to accomplish this goal. But if you want to get your feet wet with the fundamentals, I highly recommend reading my previous blog post where I defined DevSecOps.

Introduction - Need for a DevSecOps pipeline in enterprises

A CI/CD pipeline is well-known to be the backbone of any software delivery process. A CI/CD pipeline's susceptibility to security threats, however, has been quietly overlooked in recent months.

As development teams are embracing more liberal security procedures, attackers are taking advantage of the opportunity presented by the delivery/deployment phase. The major objective of the White House's Cybersecurity Improvement Executive Order 14028 was to address this very issue.

As a whole, the CI/CD part of "DevOps" has traditionally ignored security concerns. When optimizing DevOps CI/CD pipelines, security measures are frequently sacrificed for speed and scalability. Therefore, DevSecOps is based on the principle of making sure that DevOps processes are more security-focused and policy-compliant.

The vital role of 'Security' in CI/CD

Understated among the DevSecOps principles is the idea of adding "Security" to CI/CD. What many fail to realize is that DevSecOps isn't merely about moving security to the side; it's about making sure security is a top priority at every stage of development, from brainstorming and planning to coding and operations.

It has two potential introduction points:

1. 'Coding' stage by developers in the form of vulnerable code in compromised open source libraries
2. 'Build' stage by:
1. Packaging compromised artifacts
2. Using a compromised package manager
3. Improper artifact signing / injecting bad artifact that bypasses CI/CD security testing
3. 'Deployment' stage by:
1. Deploying container images to prod without proper verification/ approval
2. Not enforcing policy gates to promote a build to the next stage in CI/CD
3. Not verifying container image signature
4. 'Production'/ 'Maintenance' stage by:
1. Ineffective secrets management or by passing credentials via pipeline APIs
2. Leaving open ports or abusing privileges
3. Ineffective authorization/ authentication techniques

and more…

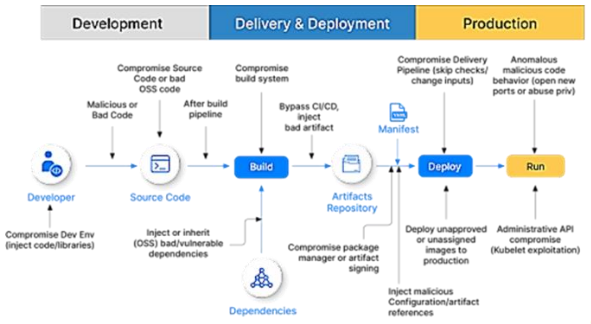
Development
Delivery & Deployment
Production
Compromise Source Code or bad OSS code
Malicious or Bad Code
After build pipeline
Compromise build system
Bypass CI/CD, inject bad artifact
Manifest
Compromise Delivery Pipeline (skip checks/ change inputs)
Anomalous malicious code behavior (open new ports or abuse priv)
Developer
Source Code
Build
Artifacts Repository
Deploy
Run
Compromise Dev Env (inject code/libraries)
Inject or inherit (OSS) bad/vulnerable dependencies
Dependencies
Compromise package manager or artifact signing
Inject malicious Configuration/artifact references
Deploy unapproved or unassigned images to production
Administrative API compromise (Kubelet exploitation)

Fig 1: *Delivery and deployment model.*

Therefore, developers may unwittingly create security flaws that could compromise company operations if adequate security checks are not in place. The application's security posture is improved and the likelihood of security events reduced when security is integrated into continuous integration and continuous delivery (CI/CD).

What is a DevSecOps Pipeline? (means to achieve CI/CD Security)

Integrating security techniques and tools into a simple CI/CD pipeline, a DevSecOps pipeline scans code, gathers threat intelligence, automates security testing, enforces policies, validates compliance, and more. The objective is to eliminate security as a distinct phase and make it an inherent component of the software development life cycle. While DevSecOps is an umbrella term for a security-focused approach to software development and delivery, a DevSecOps pipeline is more narrowly focused on making sure that continuous integration and continuous delivery pipelines are safe from outside interference. It follows that businesses which effectively implement DevSecOps pipelines boost application security, in addition to release velocity and code quality. The next step is to learn everything we can about DevSecOps pipelines.

Understanding DevSecOps Pipelines

DevSecOps pipelines have numerous moving parts; let's take a look at them all.

DevSecOps Pipeline Phases

In this blog, I shall refer to Phases in DevSecOps Pipelines as the many phases that span the full software development lifecycle. This is because the words'stages' and 'phases' are used interchangeably and lack consistency. These steps are based on the components of a "DevOps Pipeline," but they also incorporate several security procedures and technologies for apparent reasons.

Planning

During this stage of the development lifecycle, you will establish the project's security requirements and lay the groundwork for security testing. One way to find possible security concerns is to use threat modeling.

Coding

At this stage, teams need to lay out specific rules for how to develop code and make sure that everyone follows them. In order to find security flaws in source code, tools like static code analysis and peer code reviews are used.

Building

In this stage, teams are responsible for making sure that secure build methods and technologies are used for packaging and construction. It is usual practice to sign artifacts to guarantee their authenticity and integrity and to scan for dependencies to find and fix security holes in third-party libraries.

Testing

At this point, security testing becomes paramount. The following are the tasks:

- To test how well an application can withstand real-world threats, developers use dynamic application security testing (DAST).
- To examine code for vulnerabilities, one might conduct static application security testing (SAST).
- The purpose of penetration testing is to find and fix any security holes in the program.

Integration

It is important to scan all integrated components for security concerns during this step. To make sure security measures are in place, you can do things like automated security regression testing.

Deployment

During this stage, you must guarantee secure configurations during deployment and provide controls for Infrastructure as Code (IaC). One way to ensure that the deployed application is secure is to run automated deployment verification tests (DVT).

Monitoring and Incident Response

At this stage, you should set up a system to constantly check the deployed app for security issues. Crucial to the procedure is the establishment of the SRE function and the establishment of an incident response plan.

Feedback and Continuous Improvement

Gathering and analyzing security metrics to determine how well your security policies are working is the last step. You can make reports and use the comments to make your security measures better all the time.

Stages of a DevSecOps Pipeline

Given the lack of uniformity in the usage of the terms "stages" and "phases," I will be using the various activities within the "Testing" phase to illustrate the various Stages within the "Testing" phase of the DevSecOps pipeline in this blog. At each level, the emphasis is on a certain component of development or security, and the methods and tools utilized may reflect that. Among the most typical steps are:

Dynamic Application Security Testing (DAST)

DAST, or dynamic application security testing, is a crucial part of security testing because it allows businesses to find and fix application security flaws as they occur in real time. The goal, as said before, is to find security holes in live online applications by mimicking actual attacks.

Static Application Security Testing (SAST)
A security testing approach known as SAST examines an application's source code, bytecode, or binary code to find security flaws and vulnerabilities without actually running the software. It aids developers in finding and fixing security risks early on in the software development lifecycle and is performed during development, mostly at the code level.
Security Unit Testing
An important part of developing software is what's known as "Unit Testing," which involves checking the functionality of specific pieces of code to make sure they work as intended. By adding a focus on finding and fixing security-related vulnerabilities at the unit level, this method is expanded to become Security Unit Testing.
Penetration Testing
Cybersecurity testing, sometimes called pen testing or ethical hacking, is the practice of identifying and fixing security flaws in a system, network, or application by simulating an attack on it. Finding security flaws in a system and fixing them before bad guys can take advantage of them is the main objective.
Software Composition Analysis (SCA)
Analyzing and controlling software code's use of third-party and open-source libraries is known as software dependency analysis (SCA). Finding and fixing licensing problems, security holes, and other problems is the main objective when it comes to using third-party or open-source modules and packages in a project. While these are typical steps, each team will have its own unique set of challenges. The many components of a DevSecOps pipeline are the subject of the last section of this blog.
DevSecOps Pipeline Tools
Security is seamlessly integrated into the software development lifecycle with the help of DevSecOps pipelines, which utilize a range of tools. Please allow me to mention one tool for each stage of the DevSecOps process:

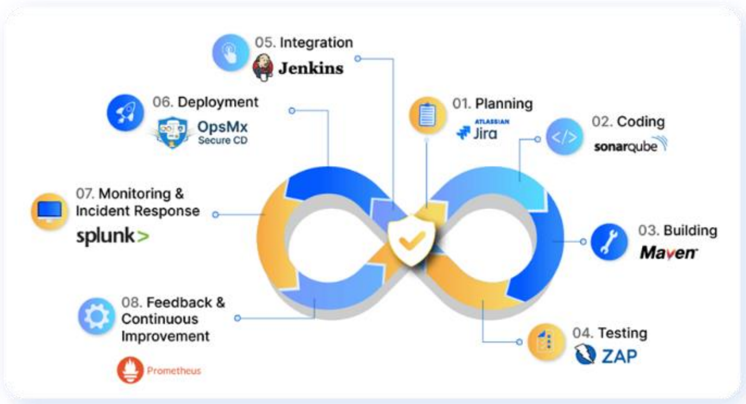

Fig 2: *Integrating system.*

Planning
Jira: Managing and planning sprints and tasks, particularly those pertaining to security, has never been easier than with this widely used issue-tracking and project-management tool.
Coding
Teams use a wide variety of tools to address various security concerns while coding. Some examples of popular ones are:
SonarQube: Utilized for offering static code analysis, continuously inspecting code quality, and ensuring security.
Checkmarx: To find and fix security flaws in source code, static application security testing (SAST) does this.
Building
Maven/ Jenkins: It is common practice to utilize a continuous integration (CI) tool like Jenkins in conjunction with the widely-used build tool Maven.
Testing
Among the many components of "DevSecOps," testing stands out. The following are examples of tools that developers utilize in a DevSecOps pipeline to scan code and prevent vulnerabilities, keeping in mind the trend of security testing moving to the left.
OWASP ZAP (Zed Attack Proxy): During dynamic application security testing (DAST), this open-source tool is utilized to identify vulnerabilities in online applications.
Burp Suite: Manual and automated security testing, including penetration testing, are both made possible by this web application testing tool.

Integration

Jenkins: This is an open-source continuous integration platform that helps with code development, testing, and deployment. Although Jenkins is mainly designed for continuous integration, it may also be utilized for continuous delivery and can be enhanced with plugins to enhance security scanning and other related tasks.

Deployment

OpsMx SecureCD: Achieving your DevSecOps goals is made easier with this software delivery/deployment tool. It aids teams in automating approvals, blocking vulnerabilities, and enforcing policy compliance; it is tailored to enterprise security and compliance.

Monitoring and Incident Response

Common monitoring, tracking, and event response tools include:

Splunk: This application is designed to help you search, monitor, and analyze data that is generated by machines, such as logs and security events.

ELK Stack (Elasticsearch, Logstash, Kibana): Identifying and keeping tabs on security issues has never been easier than with this open-source log management and analysis tool.

Feedback and Continuous Improvement

OpsMx SecureCD: Your DevOps/DevSecOps process can be enhanced once more with the help of OpsMx SecureCD. Applications' security posture, application health, deployment status, and software lifecycle improvements can all be better understood with the help of the DevSecOps dashboard.

Achieving Continuous Security: Building a DevSecOps Pipeline with OpsMx

Through its integration with numerous tools and automation of numerous procedures, OpsMx can act as a go-between in your delivery process. The many modules of OpsMx provide efficient automation during audits, approvals, and delivery verification, and they enforce security checks in a timely manner.

Vulnerability checks - pre and post deployment

There will always be vulnerabilities. They can be reported at any stage of the software development life cycle (SDLC), including development, staging, or, much worse, production.

For dynamic vulnerability (code) tracing, security products like Aquasec can integrate with OpsMx's DBOM (Delivery Bill of Materials). By enabling automated dependency validations for every build, you can extend this degree of security to dependencies and secure your whole supply chain.

Approvals and Verifications

We are cognizant of the fact that, as a result of both internal and external regulations, software delivery in organizations is notoriously difficult. Teams can be required to get approval from multiple individuals before moving the code to the next pipeline level. To guarantee authenticity and integrity, deployment verifications are necessary. DevOps (or delivery) velocity can be guaranteed with the use of OpsMx's Deployment Firewall, which automates policy gates. Additionally, it can automate policy checks, deployment verification, and rollback when necessary, as well as analyze open or broken firewall rules.

Audit & Compliance

Compliance standards and regulations can be quite strict, depending on an organization's industry of operation. Policy compliances are vital because they ensure that organizations follow all of the rules and laws. The policy-driven automatic compliance checks in the deployment firewall act as guardrails, preventing the release of code that is insecure or out of compliance. Federal Realm Mobility, Payment Card Industry, and Health Insurance Portability and Accountability Act compliance are actually embedded into our technology. Deployment Firewall users can also generate audit and attestation reports in case auditors demand evidence of responses to related issues.

DevSecOps Control Plane

Increasing transparency is currently a major issue with enterprise software delivery. Applications' security posture, health status, and deployment/delivery visibility are all severely lacking.

The DevSecOps Control Plane from OpsMx can find the application delivery and deployment process automatically, aggregate data from all DevOps and security tools, synthesize it, and correlate it. Collaboration amongst previously isolated users is made easier with OpsMx's single view across all teams, tools, and procedures. Get in touch with one of our Secure CD specialists if you need help figuring out how to put the features we covered above into action.

About OpsMx

OpsMx has been at the forefront of Secure Continuous Delivery innovation and thought leadership for quite some time. Google, Cisco, Western Union, and many more top tech businesses depend on OpsMx to provide quicker, better software. When it comes to software supply chain security, no CI/CD solution has been more pioneering than OpsMx Secure CD. Fast software delivery without compromising security is possible with OpsMx Secure CD thanks to its built-in compliance controls, automatic security assessment, and policy enforcement. Application security orchestration, correlation, and posture management are all a part of DevSecOps, which OpsMx Deploy Shield brings to your current CI/CD tools.

CONCLUSION

The topic of DevSecOps is a complicated one that has the potential to disrupt the relationship between the team and the auditors. Therefore, its deployment ought to be broken down into infractions and down infractions, with complete attention being paid to each step simultaneously. We also keep in mind that identifying vulnerabilities is only half of the task, and that giving developers the ability to quickly repair the problems that have been identified is essential. The DevSecOps methodology is a novel approach to security, and it is imperative that technologies that are specifically designed for this purpose be widely embraced. The use of DevSecOps concepts inside our continuous pipeline will reduce the likelihood of security flaws, which will ultimately lead to an increase in the level of trust that customers have in the business.

REFERENCES

[1]. L. Leite, C. Rocha, F. Kon, D. Milojicic, P. Meirelles, A survey of DevOps concepts and challenges, ACM Comput. Surv. (CSUR) 52 (6) (2019) 1–35.

[2]. Rembetsy, Michael, and Patrick McDonnell. Continuously deploying culture: Scaling culture at etsy. Velocity Europe. O'Reilly (2012).

[3]. F. Erich, C. Amrit, M. Daneva, Report: Devops Literature Review, University of Twente, Tech. Rep, 2014.

[4]. F. Erich, C. Amrit, M. Daneva, A qualitative study of DevOps usage in practice, J. Softw. Evol. Process 29 (6) (2017) e1885.

[5]. K. Carter, Francois raynaud on DevSecOps, IEEE Softw. 34 (5) (2017) 93–96.

[6]. H. Myrbakken, R. Colomo-Palacios, DevSecOps: a multivocal literature review, in: Proceedings of the International Conference on Software Process Improvement and Capability Determination, Springer, 2017, pp. 17–29.

[7]. Yasar, Hasan. Overcoming DevSecOps Challenges: A Practical Guide for All Stakeholders. Technical Report, CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH, DEFENSE TECHNICAL INFORMATION CENTER, 8725 John J. Kingman Road, Fort Belvoir, VA 22060-6218 1-800-CAL-DTIC (1-800-225-3842) United States, 2020. https://apps.dtic.mil/sti/pdfs/AD1110359.pdf.

[8]. J. Roche, Adopting DevOps practices in quality assurance, Commun. ACM 56 (11) (2013) 38–43.

[9]. M. Senapathi, J. Buchan, H. Osman, DevOps capabilities, practices, and challenges: insights from a case study, in: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, 2018, pp. 57–67.

[10]. R. Jabbari, N. bin Ali, K. Petersen, B. Tanveer, What is DevOps? A systematic mapping study on definitions and practices, in: Proceedings of the Scientific Workshop Proceedings of XP2016, 2016, pp. 1–11.

[11]. J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (Adobe Reader), Pearson Education, 2010.

[12]. E. Woods, Operational: the forgotten architectural view, IEEE Softw. 33 (3) (2016) 20–23.

[13]. J. Humble J, Farley D. Continuous delivery: reliable software releases through build, test, and deployment automation. Addison-Wesley Signature Series, Pearson Education, 27 Jul 2010 - Computers pp. 117–512.

[14]. Goldschmidt, M., McKinnon, M: Devsecops - agility with security. Technical report, Sense of Security, pp. 44 (2016).

[15]. D. Shackleford, The Devsecops Approach to Securing Your Code and Your Cloud, SANS Institute InfoSec Reading Room A DevSecOps Playbook, 2017.

[16]. T.H.-C. Hsu, Hands-On Security in DevOps: Ensure Continuous Security, Deployment, and Delivery With DevSecOps, Packt Publishing Ltd, 2018.

[17]. D. Shackleford, A Devsecops Playbook, SANS Institute InfoSec Reading Room. A DevSecOps Playbook, 2016.

[18]. J. McKay, How to use devsecops to smooth cloud deployment. SVP and Chief Technology Officer, Logicworks, IDG Communications, Inc. 2016. https://www.ne tworkworld.com/article/3041640/how-to-use-devsecops-to-smooth-cloud-de ployment.html.

[19]. V. Garousi, M. Felderer, M.V. Mantyl ¨ a, ¨ Guidelines for including grey literature and conducting multivocal literature reviews in software engineering, Inf. Softw. Technol. 106 (2019) 101–121. /02/01/2019.

[20]. B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, Version 2.3, EBSE Technical Report EBSE-2007- 01.