International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 4, Issue 1, 131-140, 2023
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V4I1P115

Original Article

Secure-by-Design Cloud Software Delivery: How DevOps and Software Teams Co-Own Security Outcomes

Sumith Thalary[1], Anvesh Katipelly[2]
[1]Sr Cloud DevOps Engineer, Rexel USA, Dallas TX.
[2]Senior Software Engineer, PayPal, Texas, USA.

Abstract *- Cloud computing has altered the current software delivery system through a platform that offers flexibility of development and enabling scalable implementation. Nevertheless, the hastening of the rate of software delivery has also posed complicated security dilemmas which cannot be addressed efficiently by the currently applied security models. The traditional methods tend to consider security to be a separate phase that follows development, leading to late vulnerability discovery, ineffective remediation, and more risky operations. Secure-by-Design paradigm has become an important means to introduce security throughout the software development life cycle (SDLC), but in that of a cloud-based DevOps setup. This scholarly article explores how software engineering teams and DevOps teams can jointly share security results with each other by means of co-determined development finesse, by establishing automation, and security control. The paper examines the potential to apply DevSecOps principles to turn around the models of traditional software development to include security mechanisms in the continuous integration and continuous deployment (CI/CD) pipelines. With automated security testing built into infrastructural security policy, vulnerability scanning, and compliance monitoring as part of system development processes, organizations can minimally decrease security vulnerabilities whilst simultaneously sustaining the pace of delivery. Another issue that is discussed in this paper is the development of cultural change in the development team so as to foster a sense of shared responsibility among the developers, the operations engineers, and the security professionals. The transition towards integrated DevSecOps units instead of remote security forces allows the vulnerabilities to be detected and removed at the initial phases of the development process. The literature review was implemented thoroughly to review prior studies regarding the DevOps security integration, cloud security architecture, and secure software engineering methods.*

The review brings out the fact that companies that embrace security-integrated pipes can enjoy better rates of vulnerability detection, remediation response, and adherence to industry standards like ISO 27001 and the NIST security models. Also, empirical evidence indicates that the organizations which have adopted secure-by-design architectures report tremendous decreases in production security occurrences relative to conventional development framework models. The proposed methodology in this study provides an ordered structure of incorporating the security controls in the various phases of DevOps such as planning, development, testing, deployment, and monitoring. The framework highlights automated threat modeling, static and dynamic testing of security, container security, infrastructure-as-code (IaC) validation, and runtime monitoring. These are mechanisms that are used to entrench security requirements in system architecture, system development practices, and system operation management. Moreover, the models and metrics of governance are proposed to gauge the maturity of security in cloud-native development environments. The study findings have shown that organizations that apply Secure-by-Design DevOps pipelines are characterized by high levels of security posture, operational resilience, and compliance preparedness. The results indicate that the vulnerability detection rates, patch response times and efficiency of incident mitigation have been determined to have improved. These enhancements show the success of shared security ownership system between operation and development teams. To summarize, the Secure-by-Design model is a paradigm shift in the field of delivering cloud software by reorganizing security as a mutual responsibility throughout DevOps ecosystems. The paper emphasizes that the need to incorporate safety measures in the development processes can greatly improve the organizational security stance in addition to agility, dependability and ease of use of cloud-centric software systems. The framework suggested can help advance the field of scholarly research and practice because it offers a systematized model of applying collaborative security governance in the contemporary cloud-based environment.

Keywords *- Devsecops, Pipelines, Secrets, Identity, Security Automation Pipeline Enforcement, Runtime Controls, Devsecops Architecture, Cloud Security Devops, Cloud IAM Integration, Security Ownership Devops,Auth Flows, Crypto Usage, Secure Apis, Secure Coding Patterns, Authentication Logic, Trust Boundaries, Secure Software Delivery, Enterprise Application Security, Shift-Left Security, Secure API Design.*

1. Introduction

1.1. Background

Cloud computing has intensely modified the process of designing, development, and implementation of modern software applications. Today, organizations are more and more entrenched with the cloud native technology and micro-service architecture, [1,2] whereby they create scalable, flexible, and highly available applications capable of responding to rapidly changing business needs. Meanwhile, the incorporation of DevOps approaches has allowed the development and operations teams to work in closer coordination and release software updates with a significantly quicker rate by applying to continuous integration and continuous deployment (CI/CD). The practices enable organizations to deliver new functions and changes regularly to enhance productivity and operation efficiency. Nevertheless, security issues are also emerging due to speed and automation of the DevOps. The blistering development cycles, distributed cloud architectures and incorporation of vast amounts of third-party elements may augment the hazard of vulnerabilities provided security endeavors are not considered collectively. Security has traditionally been considered as another stage in the software development lifecycle, and can be implemented later once the application is completed. This model security testing and validation is done at the completion of the development process and this may result in late vulnerability discovery and increasing the cost of rectifying a security problem. Such a responsive strategy will fail to work well in clouds nowadays where systems are dynamically developing and deployments happen regularly. Since the cloud systems are increasingly becoming more dynamic and decentralized, security practices cannot be left behind in modern development practices. Secure-by-Design approach resolves these issues by incorporating security principles within system architecture, development methodology, as well as operational processes of the system since the dawn of time. The inclusion of security controls in the development lifecycle can allow organizations to detect and prevent the risk before it develops and remain agile and efficient in the development of cloud based software.

1.2. Evolution of DevOps and Cloud Software Delivery

The development of the DevOps and cloud software delivery has drastically changed how the systems are being developed, deployed, and maintained. Conventional software development models were based on independent development and operation teams and this frequently resulted in a lack of communication, slowness in deployment, and poor system administration. [3,4] The necessity of speedy and more collaborative methods of software delivery has gained even greater significance as the organizations started embracing cloud computing technologies and agile approach to development processes. DevOps became a solution to the gap between development and operations teams, where they can continue working in teams, automate their work, and apply frequent software updates. As time went on, cloud platforms continued to accelerate this change by offering scalable infrastructure and automated deployment functionality as well as managed resources in a more flexible way. The further subsections illustrate the main phases in the development of DevOps and cloud-software delivery.

Fig 1: Evolution of DevOps and Cloud Software Delivery

1.2.1. Traditional Software Development and Operations

Under the conventional software development hand models, development and operations team worked in isolation with clear cut roles to perform. Writers were developers who concentrated on creating the application code and handing it over to the operations developer that was supposed to deploy and maintain the software on production lines. This segregation in most cases caused delays in communication, ineffective working processes as well as the extended time taken to release. Software releases were normally released in big and rare releases which necessitated lengthy manual code testing and manual deployment. Consequently, the process of detecting and correcting problems may consume much time resulting in high operation cost and less stability of the system.

1.2.2. Emergence of DevOps Practices

DevOps was created as a reaction to the restrictions of the conventional way of development. There is a focus on the cooperation of teams of developers and the operation department by the DevOps model where they can collaborate in the entire

lifecycle of the software. Organizations can automate application building, testing and deployment processes by following practices like Continuous Integration (CI) and Continuous Deployment (CD). Automation tools can assist with the optimization of work processes, minimize the human factor, and provide faster software updates. DevOps also promotes monitoring and feedback systems to keep on improving the performance and reliability of the system. Consequently, softwares provided by those organizations which adopt the principles of DevOps can be high-quality and delivered more efficiently as well as react to unfolding business demands.

1.2.3. Cloud-Native Software Delivery

Cloud computing has further motivated the DevOps evolution bringing a flexible and scability based infrastructure in which the application development and deployment can be undertaken. Cloud systems enable the organizations to implement applications in the distributed environments without any elaborate on-premises infrastructure requirement. Containerization, microservices models, and automation of infrastructure/systems, have allowed application developers to construct bricolage applications, which can be deployed quickly in cloud setups. Other functions that cloud-native development practices facilitate are auto-scaling, auto-resource optimization, and increased system resilience. Such features enable organizations to administer complicated applications with more efficiency and continue delivering performance and availability that are high.

1.2.4. Integration of Security in Modern DevOps

With the rising maturity of not only DevOps and cloud computing practices, the question of security became evident as to why it should be directly incorporated into the development and deployment process. The traditional security models that conducted testing at the last stages of development were not working in the DevOps environment that is fast-moving. This gave rise to the DevSecOps which adopts security practices in all the phases of the DevOps pipeline. Automated security testing, vulnerability scanning and combative surveillance tools are now in built into CI/CD pipelines to make certain that security threats are detected during the initial phases of the development lifecycle. This development guarantees that the organizations are able to assume both high speed of software delivery and high level of protection of securities in the current cloud systems.

1.3. Security Challenges in Cloud-Native Environments

Cloud-native applications have entailed emerging security issues that are not related to the previously existing on-premise infrastructures. [5] The use of microservices, containerization and distributed computing environments are common in modern day cloud applications to provide scalable and resilience services. Although flexibility and efficiency are offered by these technologies, they increase the attack surface of systems. Container vulnerabilities are one of the significant issues of cloud-native environments. Applications are packaged with their dependencies, and in cases such as this, applications can be used by attackers to gain access to the system by exploiting the dependencies, which may have old versions of a library or have known security vulnerabilities. Also, container orchestration systems and distributed resources may present a security risk unless isolation and access control measures are put in place. The other issue that is of great security concern is misconfigured cloud resources which are amongst the most frequent determinants of cloud security breaches. The storage buckets, virtual machines and networking components form part of cloud services that should be configured accurately to allow only the authorized users to access them. Nevertheless, the configuration errors may arise because of the complexity of cloud platforms and a quick deployment cycle. In illustration, publicly-available storage buckets, open ports on a network, too-liberal access controls may inadvertently make sensitive data accessible to inappropriate or unauthorized parties. Moreover, the unsecured APIs that are employed in communication among the cloud services might end up serving as entry points to the attackers in case they are not efficiently authenticated, encrypted, or rate-limited.

Chain of identity and access management is also another major challenge in cloud-native systems. Since cloud environments offer many users, many services, and automated operations, it is more difficult to control the permissions and authentication schemes. A poor identity management or wrong assignment of roles can lead to unauthorized access to important system components. Moreover, dynamic scaling capabilities of cloud systems need to have security mechanisms that can be automated to work within the ever-changing environment. Security policies need to be enacted uniformly without any manual processes as the new resources are introduced or when the scaling operations are executed. One of the most notable problems in cloud-native landscapes is that configuration vulnerabilities exist in the infrastructure-as-code (IaC) templates. IaC enables programmers to specify infrastructure setup with code, and allows them to have cloud resources automatically provisioned. These templates however, may have errors like open network ports, weak access policy and unsuitable encryption settings, this means all the vulnerability is inherited into the infrastructure. It follows that companies should deploy automated security scan systems that will examine templates and configuration files of IAC, prior to deployment. These tools aid in early identification of possible misconfigurations and thus, secure setup of cloud resources, and minimize the chance of security attacks in production setups.

2. Literature Survey

2.1. DevSecOps Security Integration

DevSecOps represents a development of the classical approach to DevOps that entails the incorporation of security practices into the software delivery and development pipeline. [6] A number of papers point out that common security testing techniques cannot be successfully used in the current environment of agile and continuous integration since they are done at the end of the development cycle. Such a late method usually leads to the identification of vulnerabilities after the code is in place and thus its repair becomes harder and more expensive. DevSecOps can resolve this issue by providing a security practice across the development life cycle, so that development, operations and security teams can work together with better efficiency. There are automated security test tools, like Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA), that are often part of Continuous Integration/Continuous Deployment (CI/CD) pipelines. These tools enable teams to scan automatically source code, running applications and third tax dependencies to remove vulnerabilities at an earlier stage of the development cycle. Consequently, DevSecOps enhances the general security position, minimizes exposure of risk, and remediation of possible risk is accelerated without hindering the development process.

2.2. Cloud Security Frameworks

Cloud security frameworks are systemic guidelines and practices that should be adopted by the organizations to exploit and mitigate security threats in the clouds. [7] Much-known frameworks like the NIST Cybersecurity Framework and the ISO 27001 provide a universal solution to undertaking security controls, performing security risk assessment, and engaged in the continuous supervision of the cloud infrastructure. The principles that are highlighted in these structures include the identity and access management, data protection, incident response and governance policies in order to secure cloud operations. The findings of the researchers have revealed that not all breaches in cloud security are the root of software vulnerabilities; they can also be configuration errors, e.g., the misconfigured storage buckets, access permissions, or poor authentication working. Therefore, the automated configuration management, incessant monitoring of compliance, and policy enforcement instruments are becoming a growing concern of modern frameworks of cloud security. Through them, organizations are able to define consistent security practices that help them increase transparency, compliance with regulations, and even resiliency against current cyber threats in the cloud.

2.3. Secure Software Development Lifecycle

Secure software development lifecycle (SSDLC) is a software model that aims at ensuring that security considerations are taken into account at each goal of software development process. [8] In contrast to the old models of development that the security testing is conducted when the product has been completed, SSDLC also is sure of implementing security in the initial planning stage, then to the design, development, testing, deployment and maintenance. The major security operations in SSDLC are threat modeling that assists in determining the possible attck vectors during the design stage, secure coding practices that reduce the prevalent vulnerabilities, automated vulnerability scanning that is aimed at identifying security vulnerabilities in code and penetration testing to demonstrate the real-life attack on the system. Empirical research studies have indicated that organizations that use SSDLC practices have significantly fewer security incidents and low cost of remediation than those which use traditional development strategies. SSDLC enhances the reliability of software, increases the defenses of a system, and encourages the security awareness culture among the development teams by actively tracking and fixing vulnerabilities within the development lifecycle.

3. Methodology

3.1. Secure DevOps Lifecycle Framework

The presented Secure DevOps Lifecycle Framework incorporates the security practices into all stages of the DevOps process. This mechanism will make sure that security is not considered as an independent stage in the development and deployment lifecycle but rather a continuous process. [9,10] The framework assists organizations in discovering and reducing vulnerabilities before it affects the organization, and the framework also ensures that the speed and efficiency of the modern DevOps environment are not compromised by adding automated tools, secure development practices, and continuous monitoring systems. The framework comprises five major stages, which include the planning and threat modeling, secure coding and development, automated security testing, secure deployment and ongoing monitoring.

Fig 2: Secure DevOps Lifecycle Framework

3.1.1. Planning and Threat Modeling

Threat modeling and planning phase aims at determining possible security threat before actual development. At this phase, development and security personnel examine the system needs, architecture, and potential attack mechanism in order to get insight into the ways the system might be breached. Identification of vulnerabilities, measures of risks, and priorities of security measures to be enforced in the development are determined through threat modeling. Early identification of threats at the very planning phase helps the organizations in developing secure architectures and set clear security requirements that will be used throughout the development process.

3.1.2. Secure Coding and Development

Secure coding and secure development phase makes sure that developers apply secure programming principles when developing applications. This involves compliance with secure coding guidelines, input data validation, and responsibility in the security management of authentication and authorization, and the avoidance of typical attacks like injection attacks or improper data processing. Secure development guidelines, code reviews and version control systems are also used by developers to ensure code integrity. By including the security awareness in the development process the vulnerability on the source is reduced and provides a better reliability and resilience to the software.

3.1.3. Automated Security Testing

The automated security testing is a necessary step in the flow of the Secure DevOps Lifecycle Framework since it provides the possibility to constantly detect the vulnerabilities throughout the development process. Continuous Integration and Continuous Deployment (CI/CD) pipelines include security testing tools that can automatically test code, dependencies and running applications to identify possible security problems. These methods are Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) which assist in identifying code bugs, runtime bugs and a third party that is insecure. Automated testing makes sure that the security checks are really frequent and continuous and do not slow down the development process.

3.1.4. Secure Deployment

Secure deployment deals with providing secure environment where applications are deployed. This stage entails deployment of security features like container security, secure configuration management, access control policy and infrastructure hardening. Deployment automation utilities assist in validating security policies and ensuring that it has configured systems properly and ready to have applications online. Through deployment security measures, the organizations are able to avoid the following regularly occurring problem, misconfigurations, unauthorized access, and exposure of sensitive data in the production environment.

3.1.5. Continuous Monitoring

The last stage of the Secure DevOps Lifecycle Framework is a continuous monitoring, which makes the applications and infrastructure secure once they are deployed. Under the production environment, monitoring tools constantly monitor the performance of the system, security incidents, and possible threats. Security analytics, logs, and alerts are used to identify abnormal activity, would be attacked, or violated policy. Incident response is also facilitated by continuous monitoring as this would make it possible to detect and fix security threats quickly. Organizations can swiftly use the visibility into system activity to respond to new threats and have a robust security posture in the long-run by keeping visibility high.

3.2. Security Automation in CI/CD Pipelines

Security automation is an essential part of the current Continuous Integration and Continuous Deployment (CI/CD) pipeline because it helps organizations to detect and resolve security weaknesses without halting or slowing the development process. [11,12] Security testing in the traditional development setting was usually manual and usually done at the last stage of the development life cycle. This method often led to late detection of the flaws which became harder and more expensive to correct. By applying automated security instruments to the CI/CD pipeline, development teams are able to maintain quality and security of their code in the entire lifecycle of software development. Automated security tests are best to prevent vulnerabilities at earlier phases of committing code, building, and testing application before effecting a significant enhancement on the security of applications, without delaying the delivery of applications. CI/CD pipelines are often configured with several security tools that facilitate vulnerable detection and risk management. SAST tools are tools that are used to scrutinize the source code at the developing and building process to determine the possible vulnerabilities which may include injection vulnerabilities, insecure pasting, and mishandling of data.

As SAST causes the source code to be executed without the application itself, it lets the developers realize the problems early during the development stage and correct them before releasing them. Dynamic Application Security Testing (DAST) in its turn is a type of testing performed on the running applications to detect vulnerabilities emerging during their runtime. DAST tools mimic real world attacks on deployed applications and assist in identifying weaknesses like authentication, configuration and input validation weaknesses. CI/CD pipelines also incorporate software composition analysis (SCA) tools with the purpose of scanning third-party libraries and open-source dependencies incorporated into the application. These can be used to detect vulnerabilities in external parts that are known and make sure that the developers are working with secure and relevant libraries. Also, Infrastructure as Code (IaC) scanners look through configuration files created to launch cloud infrastructure and identify any possible misconfigurations that would expose systems to security threats. With the aid of these automated security tools enclosed in the CI/CD pipeline, organizations will be able to build a proactive approach to security that offers accelerated development, risk assessment, and software reliability.

3.3. Risk Assessment Model

Risk assessment is a key factor in the process of securing DevOps pipelines since it assists organizations to be able to identify, assess and prioritize potential security threats that can impact the systems and applications. Vulnerabilities may emerge at various points in any given pipeline in a DevOps setup and where software is a continuous development, deployment, and integration process. [13,14] Thus, there should be a systematic risk evaluation program in order to analyze these security risks and see their degree of threat to the system. The model of risk assessment commonly applied to cybersecurity relies on the correlation of the probability of vulnerability exploitation and the extent of damage that might be caused due to the vulnerability exploitation. To put it in a simple way, simple calculation of risk is based on probability of attack that will occur and potential impact of the attack. This association may be stated as: probability x impact = risk. The concept of probability in this model is the probability that vulnerability/security weakness will be exploited by an attacker.

Likely chances of exploitation are hinged on a number of factors such as exposure of the system, attack complexity, presence of the exploit tools and the performance of the security measures in place. An illustration of this is that publicly known vulnerabilities with easy-to-exploit features are likely to be attacked. Impact, on the other hand, is the degree of harm that may well take place in case the vulnerability is exploited successfully. This harm can involve the confidentiality, integrity or availability of the system. As an example, an attack which leads to loss of data, services or hacking into sensitive information would cause high impact to the organization. Through impact and probability analysis, organizations can derive a general degree of risk of any vulnerability. Critical vulnerabilities are ones that are of high probability and high impact and are regarded as immediate risks, whilst those of lesser risk are prioritized. The model assists the DevOps teams in resource allocation, application of useable security controls and ensure consistent enhancement of the security position of the development and deployment pipeline.

3.4. Risk Assessment Model

Risk assessment is a crucial element of ensuring the security of DevOps pipelines since it aids organizations in identifying, assessing, and ranking the threats to security that can befall their systems and applications. A vulnerability may occur in any point of the pipeline in a DevOps workflow, where software is developed, integrated, and deployed continuously. [15,16] Hence, such vulnerabilities require a systematic risk evaluation framework in order to assess the severity of these vulnerabilities to the system. The risk assessment model applied to cybersecurity is established on the correlation between the probability of the vulnerability to exploitation and the magnitude of the consequence that may arise once exploitation of the same occurs. Straight-forwardly, risk can be determined as the difference between the likelihood of an attack and the effects of the given attack. This association may be defined as: Risk= Probability times Impact. Probability is used to explain this model as the chances of an vulnerability or security weakness being used successfully by an attacker. Likelihood of exploitation is determined by a number of factors such as the visibility of system, sophistication of attack, the presence of exploit tools and the efficacy of the current security mechanisms. As an illustration, the publicly known and easily exploitable vulnerabilities will have a more likely chance to be attacked. Impact on the other hand is the damage that may be caused in case the

vulnerability is exploited successfully. This could damage confidentiality, integrity or of the availability of the system. A case in point would be an attack that leads to breach of data, interference of services as well as unauthorized access to delicate data that would have a huge implication on the organization. Through assessment of probability and impact, organizations can be able to gauge the overall degree of risk on each vulnerability. High impact and high probability vulnerabilities are regarded to be critical, they need to be addressed at the earliest and the rest of the problem can be sorted according to their low risk. This risk assessment model aids the DevOps teams to be resource allocation efficient, to have proper security controls and to constantly enhance the security posture of their development and deployment lines.

4. Results and Discussion

4.1. Security Improvement Metrics

The metrics of security-enhancement are necessary to consider the effectiveness of security practices applied within the DevOps environment. [17,18] The metrics operate to assist the organizations in determining the extent to which the security processes are effective in identifying, controlling, and addressing the vulnerabilities during the software development process. Monitoring the key performance indicators (KPI) allows organizations to understand whether their security controls are operating efficiently and to determine areas that they need to improve. Security metrics also offer quantifiable data that automated security tools and secure development are aiding the enhancement of protection of applications, infrastructure, and sensitive information. Some of the most useful security enhancement measures include vulnerability detection rate, remediation time, and compliance adherence, which are also critical in consolidating the overall security posture of DevOps pipelines. Vulnerability detection rate is a measure of the capability of security tools and procedures to detect vulnerabilities that are related to the application or infrastructure. The increased rate at which it was detected implies that the built-in security systems, including automated test tools and code scanning systems, are indeed detecting the vulnerabilities prior to the implementation stages of the software into production.

Measuring this indicator assists companies in making sure that security testing is effective and can identify possible threats at the early development stage. The early-detection is also important as weaknesses identified at the early stage of development can be fixed with ease and at low cost than those identified after implementation. Remediation time is another important metric causing critical failure as it is the time that it takes to address the security vulnerabilities that have been identified. This indicator is the efficiency of development and security teams in responding and correcting security issues after they are identified. Reduced remediation durations show that there is a rapid response of teams to security notifications and the execution of remedies without a major slowdown of the development cycle. Efficient remediation procedures also enable the reduction of the time frame of attackers to exploit the vulnerabilities. Compliance adherence is the third important measure and this measure determines the level to which the organization has adhered to the set security standards, policies, as well as the rules and regulations. Compliance metrics guarantee that the security controls are anchored on the best practices and industry frameworks. With these security improvement measures that can be tracked regularly, organizations are able to continuously track their security performance, enhance their risk management strategies, and to ensure secure and reliable DevOps environment.

4.2. Security Performance Analysis

Security performance analysis is used to assess the ability to apply secure-by-design practices in DevOps environments. Through the comparison of the conventional DevOps systems with secure-by-design DevOps systems, organizations will be in a position to have a more insight on how incorporating security in the entire development life cycle will enhance the overall protection of the entire system. The security metrics below can be used to show the differences between the performance of the two approaches, especially in the key areas like vulnerability detection, patch management, reducing incidents, compliance preparedness, and automating security testing coverage.

Table 1: Security Performance Analysis

Security Metric	Traditional DevOps	Secure-by-Design DevOps
Vulnerability Detection Rate	45%	82%
Average Patch Time	60%	90%
Security Incident Reduction	35%	75%
Compliance Readiness	50%	88%
Automated Security Testing Coverage	40%	85%

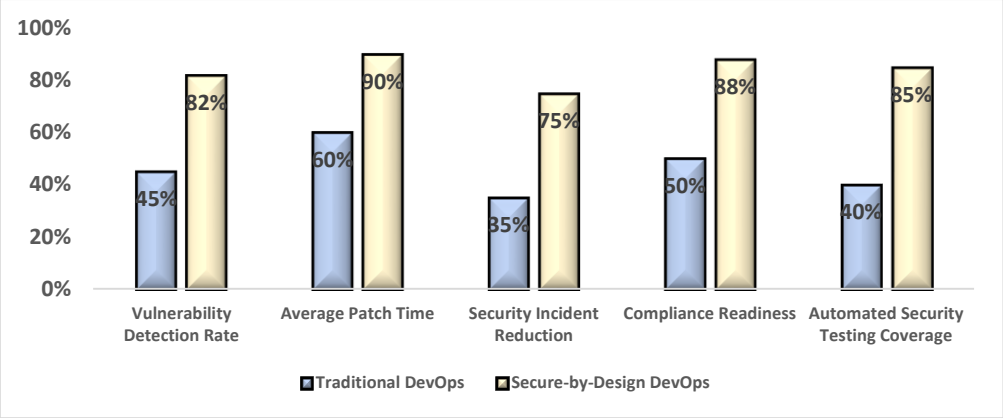

Fig 3: Security Performance Analysis

4.2.1. Vulnerability Detection Rate

Vulnerability detection rate is an action or a measurement that determines how a development pipeline is able to detect the vulnerability within the software applications. The detection rate in standard Devops is about 45% mostly due to the fact that security testing is usually done later in the development process and could be based on manual inspection methods. Conversely, secure-by-design DevOps environments have a much better detection rate of 82% because of the inclusion of automated security tools like statical and dynamical testing in CI/CD pipelines. These tools in development keep scanning code and applications to enable the teams to detect vulnerabilities at an earlier stage and minimize the chancxes of releasing vulnerable software.

4.2.2. Average Patch Time

Average patch time is the effectiveness of fixing and updating the identified vulnerabilities in the system. Conventional DevOps practices indicate efficiency patch rate of 60% because vulnerabilities are at times detected when it is too late and it takes more time to investigate and fix them. By enabling automated testing, identifying vulnerabilities early and simplifying the update process, secure-by-design DevOps raises this metric to 90%. Quickly patching the vulnerabilities makes attackers less likely to use known vulnerabilities within the time window.

4.2.3. Security Incident Reduction

Security incident reduction is used to measure the effectiveness of an organization in terms of prevention or mitigation of security breaches and attacks. The rate of reduction in the traditional Devops setting is approximately 35% meaning that a lot of the incidents happen because of the slow rate of detection and lack of preventive security measures. But in secure-by-design DevOps models, the rate of reduction is 75 percent since security is implemented at every stage of the development cycle. Constant surveillance, automatic testing and proactive threat detection will help reduce security incidents.

4.2.4. Compliance Readiness

Compliance readiness assesses the level of organizational alignment of the organization with the industry security standards, policies and regulation requirements. Conventional DevOps settings show compliance preparedness level in 50 percent because security compliance assessment is either done manually or at the final inspection phases. Conversely, secure-by-design DevOps is at the 88% readiness level through automated compliance monitoring and policy enforcement in the development pipeline. This will help in making sure that the security standards are adhered to in the course of the development and deployment.

4.2.5. Automated Security Testing Coverage

Automated security testing coverage A coverage is the measure of the degree to which security tests are performed automatically as part of the development pipeline. The coverage in the conventional DevOps systems is around 40 percent, which implies that a significant portion of the security checks can still be a matter of a manual way of testing. Secure-by-design DevOps increases this metric by a large margin to 85% through automated security scanning tools embedded in the CI/CD pipelines. This increased automation can support continuous security testing, acquire vulnerabilities faster and enhance the wholesome security of the system without slowing down development cycles.

4.3. Discussion

The security performance analysis findings indicate that considering security practices in DevOps lifecycle goes a long way in enhancing the overall security status of software development settings. The discussion of the two types of DevOps (traditional and secure-by-design) displays the importance of integrating automated security controls at every stage of the development pipeline to result in significant changes in areas of vulnerability detection, patch management, reduction of

incidents, and compliance preparedness. Conventional DevOps settings tend to focus on quick development and deployment, and thus, security can become a side matter. Consequently, the vulnerabilities might go unnoticed until later in the development lifecycle, or even after the development, raising the risk of affecting the system. The secure-by-design DevOps model, on the contrary, enforces security checks at the beginning of development, which means that vulnerabilities are identified and addressed early enough before they can cause severe harm. The fact that the vulnerability detection rate in secure-by-design DevOps settings has been increased shows that the automated security tools and continuous scanning systems are very useful in revealing the possible vulnerabilities of the software systems. The use of automated testing tools in the CI/CD pipelines helps the development teams identify security vulnerabilities at the time of coding, building, and testing.

This pre-deployment identification helps in quicker remediation and minimizes the intricacy of vulnerability repairing following deployment. The decrease in average patch time is also indicative of the fact that the organizations implementing secure DevOps can respond faster to security-related problems. Shorter Remediation assists in limiting the exposure of systems to the possible attacks and enhances the overall system reliability. The analysis has made another significant observation that the number of security incidents is reduced significantly in the case of secure development practices. Ongoing surveillance, automated threat identification and proactive security measures assist organizations to avert possible attacks before it culminates to serious attacks. Moreover, enhanced compliance preparedness demonstrates that the use of automated checks of compliance in the DevOps pipelines is a sure way of adhering to the regulatory demands and security policies at all times. On the whole, the discussion reveals that a secure-by-design DevOps implementation helps organizations to balance between fast software delivery and high-security controls, which eventually lead to more resilient and reliable software systems.

5. Conclusion

Secure-by-Design approach is one of the major developments in the current cloud software delivery as it alters the way in which security is incorporated in the software development process. Traditional models of software development have tended to view security as an independent activity done at the end of the development process with main emphasis on testing and verification before the software becomes deployed. In the present day, however, such approach is not adequate in modern cloud environments where applications are built, updated and deployed through automated pipelines with a rapid pace. The growing sophistication of cloud infrastructures and the ongoing integration and continuous deployment the pattern demand security to be built directly into the development process. Secure-by-Design applies to this challenge through redefining security as a common concern between development and operations, and security teams. Organizations can foster cooperation and coordination between these teams to make sure that security concerns are taken into account at the very beginning of software development. This paper has shown that inclusion of security control in DevOps pipelines is a much better way to ensure that organizations are able to detect and handle possible weaknesses. The development teams can identify the vulnerabilities at an earlier phase in the development cycle with the help of automated security testing tools, continuous monitoring systems, and secure coding practices. The early detection prevents the problems that may arise in the production settings before they become a risk and cause security breaches, and the risks of having to mend vulnerabilities are minimized in terms of both monetary and effort costs. Also, automated security testing in the CI/CD pipelines makes certain that security tests are conducted in a continuous and consistent manner without causing delays in the development process. Consequently, organizations are able to achieve a high level of efficiency in development and high level of security.

The suggested Secure-by-Design framework is a systematic approach to security control implementation in every phase of software development cycle, such as planning, development, testing, deployment, and monitoring. The framework assists organizations in creating a proactive approach to security instead of looking at reactive approaches after the event by introducing threat modeling, automated vulnerability scanning, infrastructure validation, and continuous monitoring practices. The results of this study point to the fact that more organizations that implement collaborative DevSecOps cultures achieve better security results and at the same time still maintain the agility and speed that modern software delivery demands. It turns into a societal culture of development of security, instead of a constraint on innovation. Moreover, the study shows that automated security testing, infrastructure configuration checking, and real-time monitoring minimizes the security risk levels in cloud-based systems. Such practices allow organizations to identify possible threats in time, implement security policies on a regular basis, and verify that security standards provided by the industry are met. In the future, even more sophisticated security automation methods could be considered in future studies to enhance DevSecOps pipelines. New technologies like threat detection programs on artificial intelligence, vulnerability analysis based on machine learning, and zero-trust security schemes can improve the security of cloud-native applications even more. Resilient cloud systems can be created by organizations by continually evolving their secure practices of developing and deploying secure technologies that could help in mitigating the dynamically changing cybersecurity environment.

References

[1] Mirtsch, M., Kinne, J., & Blind, K. (2020). Exploring the adoption of the international information security management system standard ISO/IEC 27001: a web mining-based analysis. IEEE Transactions on Engineering Management, 68(1), 87-100.

[2] Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2021). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. It Revolution.
[3] Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations. IT Revolution.
[4] McGraw, G. (2012). Software security: Building security in. Datenschutz und Datensicherheit-DuD, 36(9), 662-665.
[5] Fredj, O. B., Cheikhrouhou, O., Krichen, M., Hamam, H., & Derhab, A. (2020, November). An OWASP top ten driven survey on web application protection methods. In International Conference on Risks and Security of Internet and Systems (pp. 235-252). Cham: Springer International Publishing.
[6] Maria Bruma, L. (2021, February). Using Cloud Control Matrix to evaluate trust in cloud providers. In Proceedings of the 2021 10th International Conference on Software and Computer Applications (pp. 273-278).
[7] Singer, P. W., & Friedman, A. (2013). Cybersecurity and Cyberwar: What Everyone Needs to Know®. Oxford University Press.
[8] Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.
[9] Almorsy, M., Grundy, J., & Müller, I. (2016). An analysis of the cloud computing security problem. arXiv preprint arXiv:1609.01107.
[10] Shostack, A. (2014). Threat modeling: Designing for security. John wiley & sons.
[11] Abbasi, A. A., Abbasi, A., Shamshirband, S., Chronopoulos, A. T., Persico, V., & Pescapè, A. (2019). Software-defined cloud computing: A systematic review on latest trends and developments. Ieee Access, 7, 93294-93314.
[12] Howard, M., & Lipner, S. (2006). The security development lifecycle (Vol. 8). Redmond: Microsoft Press.
[13] Rajkumar, M., Pole, A. K., Adige, V. S., & Mahanta, P. (2016, April). DevOps culture and its impact on cloud delivery and software development. In 2016 International Conference on Advances in computing, communication, & automation (ICACCA)(Spring) (pp. 1-6). IEEE.
[14] Sarna, D. E. (2010). Implementing and developing cloud computing applications. CRC press.
[15] Chang, V., Kuo, Y. H., & Ramachandran, M. (2016). Cloud computing adoption framework: A security framework for business clouds. Future Generation Computer Systems, 57, 24-41.
[16] de Vicente Mohino, J., Bermejo Higuera, J., Bermejo Higuera, J. R., & Sicilia Montalvo, J. A. (2019). The application of a new secure software development life cycle (S-SDLC) with agile methodologies. Electronics, 8(11), 1218.
[17] Karim, N. S. A., Albuolayan, A., Saba, T., & Rehman, A. (2016). The practice of secure software development in SDLC: an investigation through existing model and a case study. Security and Communication Networks, 9(18), 5333-5345.
[18] Koskinen, A. (2019). DevSecOps: building security into the core of DevOps.
[19] Desai, R., & Nisha, T. N. (2021, July). Best practices for ensuring security in devops: A case study approach. In Journal of Physics: Conference Series (Vol. 1964, No. 4, p. 042045). IOP Publishing.
[20] Rangnau, T., Buijtenen, R. V., Fransen, F., & Turkmen, F. (2020, October). Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC) (pp. 145-154). IEEE.
[21] Antunes, N., & Vieira, M. (2014). Assessing and comparing vulnerability detection tools for web services: Benchmarking approach and examples. IEEE Transactions on Services Computing, 8(2), 269-283.
[22] Chennareddy, R. K. (2020). Engineering Intelligence Systems Using Big Data and Cloud Architectures for Modern Data Intensive Applications. International Journal of AI, BigData, Computational and Management Studies, 1(2), 41-50.
[23] Chennareddy, R. K. (2021). Designing Data and Analytics Ecosystems for High Volume Transaction Processing Applications. International Journal of AI, BigData, Computational and Management Studies, 2(2), 95-106.
[24] Sethuraman, P., & Chennareddy, R. K. (2022). Machine Learning Assisted Design of Wireless Access Systems for Reliable and Low-Latency Financial and Smart Commerce Services. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(4), 133-142.
[25] Sethuraman, P., & Chennareddy, R. K. (2022). Intelligent Vehicular Traffic Flow Prediction Using Learning-Based Spatio-Temporal Models for Data-Driven Wireless Transportation and Urban Analytics Systems. International Journal of Emerging Trends in Computer Science and Information Technology, 3(2), 111-121.
[26] Sethuraman, P. (2022). Latency-Aware Scheduling and Resource Control Algorithms for Emergency and Public Safety Wireless Networks. International Journal of Emerging Research in Engineering and Technology, 3(4), 133-140.