e-ISSN:2582 - 7219

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH IN SCIENCE, ENGINEERING AND TECHNOLOGY

Volume 5, Issue 2, February 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 5.928

International Journal Of Multidisciplinary Research In Science, Engineering and Technology (IJMRSET)

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

Shift-Left Vulnerability Scanning: Integrating Container Security Analysis into CI/CD Pipelines

Samarth Shah, Xiangbo Liang

University at Albany, Albany, NY, United States

New York University, New York, NY, United States

ABSTRACT: The shift-left paradigm lays focus on preventing the existence of vulnerabilities due to integration of security solutions with the software development life cycle (SDLC). This study investigates how shift-left vulnerability scanning methods can be applied to integrate container security analysis into CI/CD processes. By embedding security practices, such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA), into the CI/CD process, organizations can detect and mitigate vulnerabilities during code development, build, testing, and deployment stages. Trivia, Anchore and Snyk presented tools for container vulnerability scanning, and this gives a detailed description of advanced security, as well as compliance usage. Areas of concern having to do with performance, trade-offs, integration and developers are also considered. The results highlight a high degree of importance of timely detection, regular examination, and the necessity to embrace the integrated model between Development and Security teams to feature a highly efficient and secure DevOps system.

KEYWORDS: Shift-Left Security, CI/CD Pipeline, Container Vulnerability Scanning, SAST, DAST, Dev SecOps, Automated Security Tools, Compliance, Continuous Monitoring.
Text detection, Inpainting, Morphological operations, Connected component labelling.

I. INTRODUCTION

Shift left security has the following advantages for organisations and software development teams. Reduce the risks affecting your business and enhance the security of your firm through integration of protective measures and issues at the early stages of the SDLC [1]. As has been shown, in order to apply security measures early on in development, a shift-left method necessitates effective developer education [2].

Software development has a skill gap that needs to be filled, and the only way to do so is through developer education and a strong security coaching program. Since finding and correcting many faults at a later stage in the DevOps pipeline requires more time and effort, it is essential that additional Security Ambassadors come forward to ensure that more developers learn how to integrate security into their code. Also, it should also be noted that coaches can also assist in developing security measures which should start being applied in the first stages of the project [3]

While CI/CD is key to DevOps, security is often left out of the mix in more conventional DevOps settings. With CI/CD pipelines, an end-to-end pipeline may be implemented, allowing for the efficient and rapid delivery of software and software upgrades. By incorporating security procedures, such as continuous SAST and DAST procedures and security audits, into this pipeline, vulnerabilities can be found, protected, and reduced more quickly. This enables them to be found before it can be exploited and fixed without the help of security professionals.

A culture of steadily improving security practices may be nurtured by including security into CI/CD pipelines. Organisations may enhance their security posture over time by periodically scanning for vulnerabilities, analysing security data, and developing feedback loops. Organisations are at serious danger from security flaws, which may result in data breaches, monetary losses, and harm to their brand. Organisations may proactively manage and reduce these risks, protecting their assets and reputation, by integrating security measures into CI/CD pipelines [4][5].

Structure of the paper
The structure of this paper is as follows: Section II overviews shift-left vulnerability scanning in CI/CD pipelines. Section III discusses tools like SAST, DAST, and container scanning. Section IV outlines implementation challenges. Section V reviews literature and case studies, and Section VI concludes with future directions.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

II. KEY CONCEPTS AND TECHNOLOGIES

Security is given priority at the outset of program development via shift-left vulnerability screening. Developers may identify vulnerabilities throughout the code development, build, and deployment phases due to the integration of security into CI/CD procedures. The majority of vulnerabilities are discovered using container, dynamic, and static scanning. Implementing security is more organic due to automated tools, security rules, and developer training [6]. Shift-Left Security relies on technology, and by integrating security testing early in the SDLC, vulnerabilities should be identified and fixed throughout software design and development [7]. Similarly, automated security technologies (SAST, DAST, and container scanners) are used in CI/CD pipelines to provide safe and legal development processes for software delivery, deployment, and continuous integration.

Shift-left security

Software and code security assurance procedures are implemented as early as feasible during the SDLC through the practice of shift-left security. Shifting to the left indicates that security testing will be conducted earlier in the software and application development phase. In traditional DevOps, the various stages would flow like this: Plan > Code > Build > Test > Deploy > Monitor. Security concerns are tackled early on by "shifting left," which aids in finding and fixing security holes before it escalate into expensive problems during development or after deployment [8].

Shift Left Security Tools

The integration of automated tools into the workflow is a critical step in the effective shift of security left, despite the fact that developers may find it difficult to implement and learn new tools. A number of the most critical instruments required for the implementation of shift left security policies are as follows:

- **Static Application Security Testing (SAST) tools:** Code injection [9], buffer overflows, and unsafe coding techniques are among the security vulnerabilities that this tool may detect by analysing the source code or producing binaries.
- **Dynamic Application Security Testing (DAST) tools:** This program sends malicious queries and analyses the answers to find security flaws in running apps. In order to find vulnerabilities like SQL injection, failed authentication, and cross-site scripting (XSS), that mimic real-world assaults.
- **Interactive Application Security Testing (IAST) tools:** The program integrates features from both SAST and DAST to analyse application code while it is running, allowing for real-time vulnerability detection and a better understanding of application behaviour and security concerns [10].
- **Software Composition Analysis (SCA) tools:** In order to lessen the likelihood of security issues caused by out-of-date or insecure third-party libraries and components, this utility scans software dependencies.
- **Static Code Analysis tools:** This program finds security flaws, including injection vulnerabilities and unsafe coding practices, in real-time while developers write code and give them quick feedback on the code's quality and security.

CI / CD Pipeline

Through the incorporation of automation into software development processes, CI/CD facilitate the rapid, repeatable, and safe delivery of software updates to users [11]. Since code is committed at one end of CI/CD and tested at each step (source, build, staging, and production) before being ready for release, the process may be thought of as a pipeline. An ongoing practice is to construct a CI/CD pipeline. The CI/CD method, similar to the software being developed, uses iterative ways to analyse data and wait for feedback in order to perfect itself [12].

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

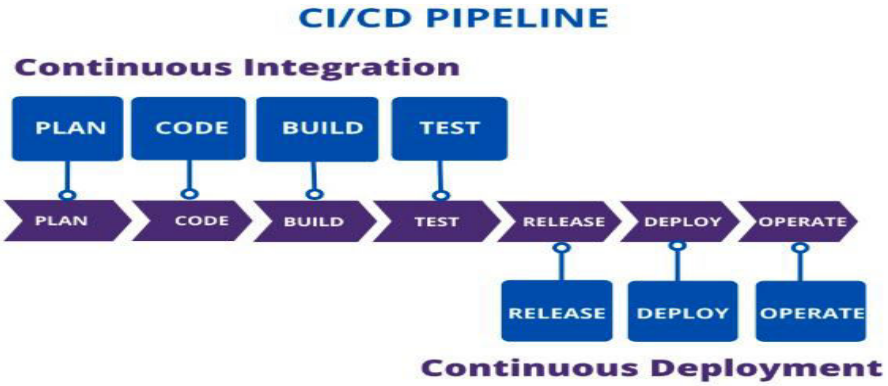

Fig.1. The CI/CD Pipeline

CI/CD pipeline security is the integration of security into CI/CD processes, bringing the same types of efficiency and consistency to security processes that CI/CD itself offers to development operations. CI/CD pipeline, as depicted in Figure 1:

- **Plan:** The process begins with planning the project, defining its requirements, features, and a roadmap for its development.
- **Code:** While creating the software, the developers code it following the plan usually in controlled versions through methods such as Git.
- **Build:** Thereafter the code is compiled into run time or application where the actual program can be run. This step is usually performed with some automated build tools.
- **Test:** To enhance the quality of the built software it is tested based on the standardised quality standards. Such tests may include unit tests, integration tests and other tests that are conducted in the process.
- **Release:** If the software meets all the tests it is built and made ready for release to the market. This might include generating installers, or even container image specifications.
- **Deploy:** The packaged software is then deployed to a production or staging environment.
- **Operate:** Monitoring and managing is the procedure which happens in the production environment after the software has been released. These are things like scaling, and configuration across systems and responding to any problems that may occur during the process.
- **Monitor:** It is therefore important that the authority in charge of the software will monitor it continuously with an aim of controlling for any arising issues after the software has been released onto the market.

III. FUNDAMENTAL OF CONTAINER VULNERABILITY SCANNING

There are a number of reasons why container vulnerability scanning is critical for finding and fixing security holes in containerised apps. Containers can be used and are in use today thus requiring a surety cover for containers due to their importance in the current software development and deployment architecture. Spotting and fixing security holes in containerised applications requires regular container vulnerability assessment. The process of identifying security issues concerns the evaluation of the container and its root base, application source code, and all the dependencies concerning any existing vulnerabilities.

Common Container Vulnerabilities and Their Impact

The subject of container security is closely associated with the types of vulnerabilities which exist, and there are a lot of them; moreover, container CVEs are described by such organisations as MITRE and NVD.

Base Image Vulnerabilities

A significant proportion of containers are derived from application relationships associated with simple images that may contain vulnerabilities. Base images are generally rarely outdated nor do they get left unpatched, and so they contain exploitable known vulnerabilities. To detect any outdated or vulnerable parts of the application, the container vulnerability scanner then aligns these basic images to such lists as CVE.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

Application Code Vulnerabilities
Application vulnerabilities may be caused by misuse of configuration settings, out-of-date libraries, or bad coding. These holes in the code can be exploited by hackers. The presence of these may cause serious problems, such as unauthorised access or data breaches.

Dependency Vulnerabilities
Commonly used by software development initiatives are external libraries and pieces of open-source code. These dependencies have the potential to cause further issues.

Container Runtime Vulnerabilities
The shared kernel concept poses a significant threat to containers because it allows several containers on a single host to share an operating system kernel. There might be major security breaches if one of them is hacked and then it affects the others or the host. In addition, these hazards might be amplified by setup problems. Exposing ports to the internet by accident or using weak authentication are common blunders.

Supply Chain Attacks
Attackers may use well-known photos to infect them with malware and access computers. When hackers figure out how to infiltrate network components or introduce harmful code into software, supply chain assaults occur. These assaults focus on the creation and dissemination of container images [13]. It also tries to find ways into the supply chain, such as flaws in apps or updates, and take advantage of such to get important digital resources.

Importance of Container Vulnerability Scanning:

The Importance of Container Vulnerability Scanning Enhancing Security and Compliance in Modern Development. An essential reason to check containers for vulnerabilities is as follows:

- **Early detection of vulnerabilities:** While developing and deploying container images, organisations might find security concerns by scanning them for vulnerabilities [14]. Organisations may lessen the likelihood of security breaches by finding vulnerabilities early and fixing them before deploying them to production settings.
- **Compliance requirements:** Security controls and procedures are often a need in light of regulatory compliance rules; many organisations are subjected to. To ensure that container images are secure and free of known vulnerabilities, organisations may use container vulnerability scanning to achieve these criteria. Discover more information in our comprehensive guide on container compliance.
- **Minimizing attack surface:** Attack vectors may be minimised by identifying and stripping out unnecessary parts, packages, and dependencies that might be risky through the scanning of container images for vulnerability.
- **Continuous security:** Software is built, tested, and released quickly under current DevOps and CI/CD techniques. Container vulnerability scanning keeps security current with the rapid release cycles by screening container images at each level of the development process, ensuring continuous protection [15].
- **Secure dependencies:** Container images rely on third-party libraries, packages and base images which may contain various potential vulnerabilities. To detect and mitigate risks associated with these dependencies, organisations may want to inspect container images.
- **Enhanced security posture:** Organisations may enhance the security situation and protect itself from new Identified security risks to containerised applications by occasionally scanning the container images for vulnerabilities [16].

IV. CHALLENGES IN IMPLEMENTING SHIFT-LEFT VULNERABILITY SCANNING

There are some challenging phases when we implement the shift-left vulnerability scanning given below:

Performance Trade-offs in Scanning Tools

- **Impact on Build Times:** Some vulnerability scanning tools, for example, may take quite a bit of computing, which in turn slows the CI/CD process. When multiple scans are integrated into the workflow, the build and deployment cycles might be delayed, impacting delivery timelines in agile environments.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

- **Resource Utilization:** Scanning large container images or complex applications can consume substantial CPU and memory resources, leading to performance bottlenecks, especially in shared environments.
- **Optimization vs. Thoroughness:** A trade-off exists between the depth of scans (thorough analysis) and the speed of execution. Faster scans might skip certain checks, while more comprehensive scans may introduce delays.

Integration Challenges with Existing CI/CD Platforms

- **Compatibility:** Many organizations use diverse Tool CI/CD platforms (e.g., Jenkins, GitLab, GitHub Actions) that may not seamlessly integrate with all vulnerability scanning tools [17]. Ensuring compatibility and proper configuration can be time-consuming.
- **Pipeline Complexity:** Incorporating security scans into a CI/CD pipeline may require reconfiguring existing workflows, which can disrupt established processes. This complexity increases with hybrid or multi-cloud environments.
- **Version Control and Dependencies**: Different components of the pipeline (e.g., plugins, libraries) may require frequent updates to remain compatible with scanning tools, adding maintenance overhead.

Addressing False Positives and False Negatives

- **False Positives:** Scanning tools sometimes flag non-critical or already remediated issues as vulnerabilities [18]. This can overwhelm developers with unnecessary alerts, leading to "alert fatigue" and reduced trust in the tools.
- **False Negatives:** Missed vulnerabilities (false negatives) are a critical issue as it leaves applications exposed to potential threats. These often arise due to incomplete vulnerability databases or lack of updates in scanning algorithms [19].
- **Balancing Accuracy and Usability:** Evaluating the rates of false positivity and false negativity, and choosing optimal values for these parameters, is an infinite problem, and scanning tools need to be updated frequently.

Ensuring Developer Adoption and Minimizing Disruptions to Workflows

- **Developer Resistance:** It is noteworthy that developers may experience security scanning as an encumbrance, which hinders their activity, provided that scans are invasive, or alerts are frequent.
- **Learning Curve:** There should be open access to updated scanning tools and process changes since the integration can entail some training, which can be tiresome for teams that may not have a clue concerning secure coding.
- **Workflow Interruption:** Some of the scanning tools that are not very well integrated may cause constant build failures or slow down the overall deployment. Additionally, it can create frustration and hinder the implementation of shift-left processes.
- **Cultural Shift:** Transferring security tasks up to the initial steps of the engineering process depends on the security culture to shift the perception from the security silos.

V. INTEGRATING CONTAINER SECURITY ANALYSIS INTO CI/CD PIPELINES

Container security analysis, in particular, should be included in the CI/CD process to ensure continuous application security throughout SDLC. Given the current trend of using containers and microservices, the treatment of vulnerabilities in environments involving containers has been a priority. Container security in CI/CD pipeline minimises possible risks before going live through prevention and protection of applications. The quick software and application deployment offer advantages; however, embedding security tends to be complicated. Security requirements can be plugged in into the CI/CD pipeline by incorporating container security analysis, which can help mitigate risks at different stages of the software delivery lifecycle.

Significance of Container Security

To begin with, containers can improve an organisation’s capability and ability; however, indeed, there are security risks as well. Below are the challenges:

- **Vulnerable Images:** Are based libraries or dependencies written to avoid backtracking any old vulnerabilities?

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

- **Misconfigurations:** If an Unprivileged or insecure Docker file or Kubernetes is used that can lead to data exposure and privilege escalation, there's a risk.
- **Runtime Threats:** Any unauthorised exploitation that intercepts the runtime of a container or malicious activity occurring within it.

Security Integration Key Stages

Inside the CI/CD pipeline, security can be integrated at different levels in order to provide secure boundaries at both the ends.

Pre-Build (Source Code Management)

- **Static Code Analysis:** At what point so we start the scanning process? We can use analysis tools that scan the source code for vulnerabilities before the actual build [20].
- **Audit Configuration Files:** Are the best practices being followed as far as configuring Dockerfiles and Kubernetes manifests or not? If not then don't evade being veracious to audit them.

Build Stage

- **Image Scanning:** From tools such as Trivy or Anchore conduct scans against container images to ensure no vulnerabilities or outdated dependencies exist.
- **Dependency Security:** Ensure risks against open-source libraries and frameworks are being actively scanned.
- **Configuration Validation:** Examine the contents of Docker files to assess for insecure defaults including but not limited to exposing sensitive ports or running root containers.

Testing Stage

- **Dynamic Application Security Testing (DAST):** Address DAST by replicating real-world attacks in covered zones during application development.
- **Security Integration Testing:** To test how securely containers can be integrated with APIs and other systems, use integration testing focusing on security.
- **Compliance Checks:** To examine whether the procedures followed were compliant, determine CIS Benchmarks or NIST guidelines compliance.

Deployment Stage

- **Registry Scanning:** Carry out scans against container images retrieved from Docker hub, windows server, AWS ECR or Azure ACR register TIFF images.
- **Policy Enforcement:** Policies concerning security should be enforced using OPA tools or Kubernetes Admission Controllers before the deployment.

Post-Deployment (Runtime Security)

- **Runtime Monitoring:** Tools of Falco or Aqua Security live containers and detect unusual activity.
- **Threat Detection**: For malicious activities, establish intrusion detection and alerting in real-time.

Benefits of a Shift-Left Approach

The application of a Shift Left Security approach allows security to be incorporated right from the beginning of the CI/CD pipeline integration processes.

- **Early Detection:** Identifies vulnerabilities early, reducing the cost and effort of remediation.
- **Improved Collaboration:** Facilitates collaboration between development, operations, and security teams.
- **Continuous Compliance:** Ensures adherence to security and regulatory standards throughout the development lifecycle.

Tools for Container Security Integration

There are several tools available to support container security in CI/CD pipelines:
Trivia: Lightweight scanner for container image vulnerabilities and misconfigurations.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

Anchore: Provides deep scanning and compliance checks for container images.
Snyk: Scans dependencies and Docker images for vulnerabilities.
Falco: Monitors container runtime behaviour for anomalies [21].
Aqua Security: Offers comprehensive security for the entire container lifecycle.

VI. LITERATURE REVIEW

This section provides a comprehensive review of the literature on Shift-Left Vulnerability Scanning: Integrating Container Security Analysis into CI/CD Pipelines, with a summarised overview presented in Table I.

In this study, Kaulgud et al., (2016) establishing a taxonomy for organizing testing kinds, analytics, and data needs; and conducting continuous, shift-right testing. 'Shift left' has been the guiding principle for software testing optimization for a long time. This approach prioritizes automating testing as early as feasible in the SDLC. In order to do this, testing has to "shift right," or go beyond the deployment boundary, and embrace a data-driven strategy that makes use of production data collected after release and input from customers[22].

In this study, Wang et al., (2019) the vulnerability scanning method is described in detail. Secondly, the vulnerability scanning technique is examined in detail, with an emphasis on how it is put into practice. At long last, the vulnerability scanning system's primary goals have been accomplished, and the scanning findings have validated the research's viability. Web application vulnerability screening may be made more efficient with the use of an automated scanning system, as opposed to the laborious, time-consuming, and ineffective manual vulnerability detection methods[23].

In this study, Wang and Yang, (2017) examine the most recent open-source vulnerability scanning technologies and their present levels of sophistication. Our lab design incorporates a virtual lab environment. They provide a detailed presentation of the hands-on labs that they developed using OpenVAS, a vulnerability detection tool. It would be challenging to implement effective network defence to thwart real-world invaders without first comprehending computer system weaknesses. Thus, a crucial component of an effective cybersecurity curriculum is the instruction of ethical hacking as well as vulnerability scanning[24].

In this study, Pothula et al. (2019) make use of examples to demonstrate security flaws in general and build on a common knowledge of container security. A comprehensive evaluation of the effects of vulnerabilities on container security is offered, including the controls, best practices, and methods for protecting the system. With the advent of container technology, the cloud business has entered a new operational phase, and the elimination of security vulnerabilities is of the utmost importance for both public and private clouds, since attacks may start inside the cloud itself [25].

In this study, Singh et al. (2019) reviewing and contrasting several continuous integration (CI) and delivery (CD) solutions with respect to many criteria, such as post-deployment performance monitoring, integration of pipelines, interoperability with the cloud, and server monitoring. However, managing and deploying these services gets more challenging as the number of microservices increases. A microservice architecture is a method for improving cloud applications that allows large-scale businesses to grow their application according to demand [17].

In this study, Agarwal, Gupta and Choudhury, (2018)offers a range of techniques and resources that can make up a successful CD/CI pipeline. Culture, automation, measurement, and sharing (CAMS) are the cornerstones of DevOps. Its continuous approach to software delivery, continuous integration (CI), and continuous deployment (CD) is what is making it more and more popular. With continuous integration (CI), the developer must commit the code multiple times throughout the day, after which it will automatically build and test and provide the developer with instant feedback if a defect is found[26].

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

Table I: Integrating Container Security Analysis into CI/CD Pipelines with Shift-Left Vulnerability Scanning

Reference	Key Topic	Focus Area	Findings/Insights
[22]	Continuous Testing and Shift Right Testing	Taxonomy of testing types, data-driven approaches, and testing optimization	Emphasises a hybrid approach of shift-left and shift-right strategies for early testing and leveraging post-release data for optimization.
[23]	Working Principles of Vulnerability Scanning	Implementation and automation of vulnerability scanning systems	Automated systems enhance efficiency and effectiveness of vulnerability scanning compared to traditional manual approaches.
[24]	Open-Source Vulnerability Scanning Tools	Virtual lab environments and hands-on learning with OpenVAS	Teaching vulnerability scanning and ethical hacking is essential for successful network defence in cybersecurity education.
[25]	Container Security and Vulnerability Analysis	Impact and controls for container security vulnerabilities	Identifies critical vulnerabilities in containerised environments and proposes controls and best practices for securing both public and private clouds.
[17]	CI/CD Tools and Microservice Architecture	Integration, performance monitoring, and cloud compatibility	Highlights challenges in managing and deploying microservices in CI/CD pipelines, emphasising scalability and monitoring.
[26]	Effective CI/CD Pipeline Methodologies	Emphasis on integrating automation, measurement, and feedback mechanisms in DevOps processes.	Highlights the importance of culture (CAMS) and continuous approaches (CI/CD) for software delivery. Recommends frequent code commits, automated builds/tests, and immediate bug feedback to enhance efficiency.

VII. CONCLUSION AND FUTURE WORK

The integration of shift-left vulnerability scanning into CI/CD pipelines represents a significant advancement in enhancing software security. By embedding security measures early in the development lifecycle, organisations can proactively identify and address vulnerabilities, reducing risks and improving compliance with industry standards. This review highlights the growing importance of adopting a shift-left approach to vulnerability scanning by integrating container security analysis into CI/CD pipelines. By analysing the current state of tools, methodologies, and practices, the paper underscores the critical role of early vulnerability detection in minimising risks, reducing costs, and ensuring compliance with security standards. The review identifies key challenges, including tool performance trade-offs, false positives, and integration complexities, while emphasising the benefits of fostering collaboration between development and security teams. Future research should explore the development of more efficient and intelligent scanning tools that minimise performance bottlenecks and reduce false positives and negatives. Additionally, integrating machine learning and artificial intelligence (AI) into security tools could enable more predictive and adaptive vulnerability detection.

REFERENCES

[1]S. Georgiou, S. Rizou, and D. Spinellis, “Software development lifecycle for energy efficiency: Techniques and tools,” ACM Comput. Surv., 2019, doi: 10.1145/3337773.
[2]A. Othman, N. A. Kamarul Zaman, and R. Swamy, “Experiences in Shift Left Test Approach,” Int. J. Emerg. Technol. Adv. Eng., 2019.
[3]D. A. Meedeniya, I. D. Rubasinghe, and I. Perera, “Traceability establishment and visualisation of Software Artifacts in DevOps Practice: A survey,” Int. J. Adv. Comput. Sci. Appl., 2019, doi: 10.14569/ijacsa.2019.0100711.
[4]M. Shahin, M. Ali Babar, and L. Zhu, “Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices,” 2017. doi: 10.1109/ACCESS.2017.2685629.

| ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 5.928|

| Volume 5, Issue 2, February 2022 |

| DOI:10.15680/IJMRSET.2022.0502001 |

[5]A. F. Nogueira, J. C. B. Ribeiro, M. Zenha-Rela, and A. Craske, "Improving la Redoute's CI/CD pipeline and DevOps processes by applying machine learning techniques," in Proceedings - 2018 International Conference on the Quality of Information and Communications Technology, QUATIC 2018, 2018. doi: 10.1109/QUATIC.2018.00050.
[6]S. Bairwa, B. Mewara, and J. Gajrani, "Vulnerability Scanners-A Proactive Approach To Assess Web Application Security," Int. J. Comput. Sci. Appl., vol. 4, 2014, doi: 10.5121/ijcsa.2014.4111.
[7]J. Q. Gandhi Krishna, "Implementation Problems Facing Network Function Virtualization and Solutions," IARIA, pp. 70–76, 2018.
[8]S. K. Choi, C. H. Yang, and J. Kwak, "System hardening and security monitoring for IoT devices to mitigate IoT security vulnerabilities and threats," KSII Trans. Internet Inf. Syst., 2018, doi: 10.3837/tiis.2018.02.022.
[9]B. Aloraini, M. Nagappan, D. M. German, S. Hayashi, and Y. Higo, "An empirical study of security warnings from static application security testing tools," J. Syst. Softw., 2019, doi: 10.1016/j.jss.2019.110427.
[10]Y. Pan, "Interactive application security testing," in Proceedings - 2019 International Conference on Smart Grid and Electrical Automation, ICSGEA 2019, 2019. doi: 10.1109/ICSGEA.2019.00131.
[11]S. Bobrovskis and A. Jurenoks, "A survey of continuous integration, continuous delivery and continuous deployment," in CEUR Workshop Proceedings, 2018.
[12]Y. Ska and R. Publications, "A STUDY AND ANALYSIS OF CONTINUOUS DELIVERY, CONTINUOUS INTEGRATION IN SOFTWARE DEVELOPMENT ENVIRONMENT," SSRN Electron. J., vol. 6, pp. 96–107, 2019.
[13]W. Kangjin, Y. Yong, L. Ying, L. Hanmei, and M. Lin, "FID: A faster image distribution system for docker platform," in Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017, 2017. doi: 10.1109/FAS-W.2017.147.
[14]M. Yi, X. Xu, and L. Xu, "An Intelligent Communication Warning Vulnerability Detection Algorithm Based on IoT Technology," IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2953075.
[15]R. D. Deepak and P. Swarnalatha, "Continuous Integration-Continuous Security-Continuous Deployment Pipeline Automation for Application Software (CI-CS-CD)," Int. J. Comput. Sci. Softw. Eng., 2019.
[16]B. Dickinson and D. Wilkinson, "Securing industrial systems in a digital world," APPEA J., 2019, doi: 10.1071/aj18264.
[17]C. Singh, N. S. Gaba, M. Kaur, and B. Kaur, "Comparison of different CI/CD Tools integrated with cloud platform," in Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019, 2019. doi: 10.1109/CONFLUENCE.2019.8776985.
[18]D. Colquhoun, "The False Positive Risk: A Proposal Concerning What to Do About p-Values," Am. Stat., 2019, doi: 10.1080/00031305.2018.1529622.
[19]G. Guillera-Arroita, J. J. Lahoz-Monfort, A. R. van Rooyen, A. R. Weeks, and R. Tingley, "Dealing with false-positive and false-negative errors about species occurrence at multiple levels," Methods Ecol. Evol., 2017, doi: 10.1111/2041-210X.12743.
[20]K. Goseva-Popstojanova and A. Perhinschi, "On the capability of static code analysis to detect security vulnerabilities," Inf. Softw. Technol., 2015, doi: 10.1016/j.infsof.2015.08.002.
[21]H. Saito, H.-C. C. Lee, and C.-Y. Wu, DevOps with Kubernetes : accelerating software delivery with container orchestrators. 2017.
[22]V. Kaulgud, A. Saxena, S. Podder, V. S. Sharma, and C. Dinakar, "Shifting testing beyond the deployment boundary," in Proceedings - International Workshop on Continuous Software Evolution and Delivery, CSED 2016, 2016. doi: 10.1145/2896941.2896954.
[23]B. Wang, L. Liu, F. Li, J. Zhang, T. Chen, and Z. Zou, "Research on Web Application Security Vulnerability Scanning Technology," in Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2019, 2019. doi: 10.1109/IAEAC47372.2019.8997964.
[24]Y. Wang and J. Yang, "Ethical hacking and network defence: Choose your best network vulnerability scanning tool," in Proceedings - 31st IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2017, 2017. doi: 10.1109/WAINA.2017.39.
[25]D. R. Pothula, K. M. Kumar, and S. Kumar, "Run Time Container Security Hardening Using A Proposed Model of Security Control Map," in 2019 Global Conference for Advancement in Technology, GCAT 2019, 2019. doi: 10.1109/GCAT47503.2019.8978433.
[26]A. Agarwal, S. Gupta, and T. Choudhury, "Continuous and Integrated Software Development using DevOps," 2018. doi: 10.1109/icacce.2018.8458052.

INNO SPACE
SJIF Scientific Journal Impact Factor
Impact Factor: 5.928

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

9710 583 466 9710 583 466 ijmrset@gmail.com

www.ijmrset.com