IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Issue 10, October 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

Shift Left Security: An Approach to Vulnerability Assessment in Continuous Integration and Delivery Pipelines

Varadharaj Krishnan

Independent Researcher, Seattle, Washington, USA

ABSTRACT: This paper explores the concept of shift-left security and an approach to implementing it. It delves into how to implement security measures into the early stages of the software development lifecycle (SDLC), i.e., within the different stages of continuous integration and delivery (CI/CD) pipelines. As modern application development increasingly adopts CI/CD practices, integrating security checks at every phase of the pipeline is vital to detecting vulnerabilities early and mitigating risks. By providing a comprehensive, stage-by-stage breakdown, this paper outlines actionable strategies, tools, and best practices to ensure vulnerabilities are addressed before they reach production. It serves as a practical guide, offering a blueprint for organizations to follow and adapt, ensuring that security is an integral part of the software development process from the outset. This approach not only enhances the security posture of applications but also reduces the costs associated with post-deployment fixes, providing a scalable, efficient method for improving overall security in modern development environments.

KEYWORDS: Shift-Left Security, Cybersecurity, Vulnerability Management, Secure CI/CD, DevSecOps.

I. INTRODUCTION

In modern application development, security is an essential component of the entire software development life cycle. Security is no longer a mere final checkpoint prior to deployment to production. Shift-left security refers to the implementation of code and software security assurance measures at the earliest stages of the software development lifecycle (SDLC). By performing security assessments on code, infrastructure elements, and applications, developers can address vulnerabilities and misconfigurations at the earliest stages of development. If we visualize the software development lifecycle as a process starting from the left with coding as the first stage and ending on the right with a running application, Shift-Left Security refers to implementing security controls and assessments during the early stages of continuous integration and continuous delivery pipeline (CI/CD) before the code is deployed for production. CI/CD is now the de facto methodology followed as part of modern DevOps practice; it enables teams to merge code into a code repository continuously, and with a sequence of automation, the code is built and deployed to the target environment, enabling developers to see the results quickly. As software becomes more complex, the surface area for vulnerabilities too increases. Vulnerabilities are weaknesses in the software that can be exploited by malicious actors to compromise security, leading to breaches, data theft, and other significant risks. By catching vulnerabilities early, development teams can fix them before they are propagated into later stages of development or into production, reducing the risk of costly post-deployment fixes and improving the overall security of the application. This paper aims to provide a practical approach to how and what to do shift-left.

II. BACKGROUND

With a traditional approach to vulnerability assessment, assessment typically happens after software development is completed; this poses significant challenges. Security assessments done late in the lifecycle are reactive, resulting in the discovery of vulnerabilities where that are expensive and time-consuming to address. By the time a vulnerability is identified, it can be deeply embedded into the design or critical part of the code or dependency, which cannot be modified easily or would need extensive regression testing. Any attempt to do manual security reviews and audits will slow down the whole process. The slow remediation cycle with traditional vulnerability assessment will also mean that vulnerabilities may not be addressed quickly, and the application might have prolonged exposure. The need for Shift Left Security arises from the deficiencies of traditional security practices. By incorporating security checks earlier, especially within every stage of the CI pipelines, we have an opportunity to take proactive measures. Embedding security assessments at multiple points within the CI/CD pipeline improves the chances of detecting vulnerabilities earlier and ensures that security testing is continuous and automated.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

III. CONTINUOUS INTEGRATION AND DELIVERY PIPELINE

The disciplines of continuous integration and continuous delivery are combined in CI/CD, which is a subset of DevOps. CI/CD automates the build, test, and deploy phases, as well as infrastructure provisioning, thereby alleviating the need for manual human intervention in the process of transitioning new code from a commit to production. Development teams can implement modifications to code that are subsequently automatically tested and deployed through a CI/CD infrastructure. Correct implementation of CI/CD reduces downtime and speeds up code releases. Additionally, CI/CD facilitates more rapid feedback cycles with stakeholders, guaranteeing that the final product closely aligns with user expectations. Overall, it is a fundamental practice for any team that is striving to achieve high-quality, high-speed software development.

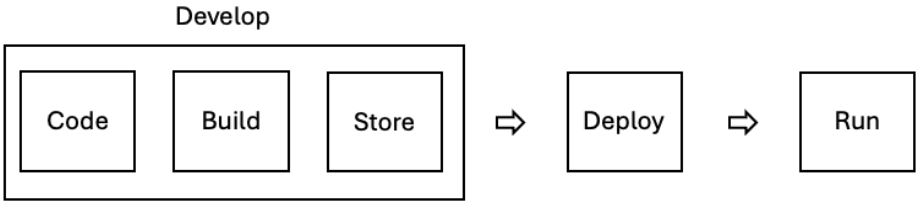

Figure 1 Simple illustration of component of CI/CD pipeline.

Figure 1 shows a simple illustration of the components of a CI/CD pipeline. Additionally, it represents major steps in the software development lifecycle. There are various tools and technologies used for each step of the build pipeline. In modern-day container-based software development, there are few established patterns and toolsets. In this paper, we will consider the container-based application build and deployment and how to shift left security for such an environment.

IV. SHIFT-LEFT APPROACH

4.1 Develop

The development stage is primarily made up of three different substages, namely Code, Build, and Store. Modern-day developers code through sophisticated IDEs (Integrated Development Environments). These IDE are powerful tools in assisting the developer to be productive and improve velocity and code. When thinking about shifting left, we can start from the IDE, where the code is written for the first time. There are several linting and code quality check tools available. Similarly, static code scanning tools can be configured directly within the IDE to prompt for problematic code snippets during the build time. These smart IDEs have a complete feature set to perform local builds as well. Code scanning can be embedded in that local build. The local build is different from the CI/CD build stage. Depending on the type of application or code that is developed, there are options to shift left. In the case of cloud development, developers develop Infrastructure as Code (IaC). These can be checked for misconfiguration right within the IDE using the plugins made available by the security solution provider, for example, Wiz.io or Aqua. If it is a container-based development, local images can be scanned within the IDE using similar plugins made available by the security solutions providers.

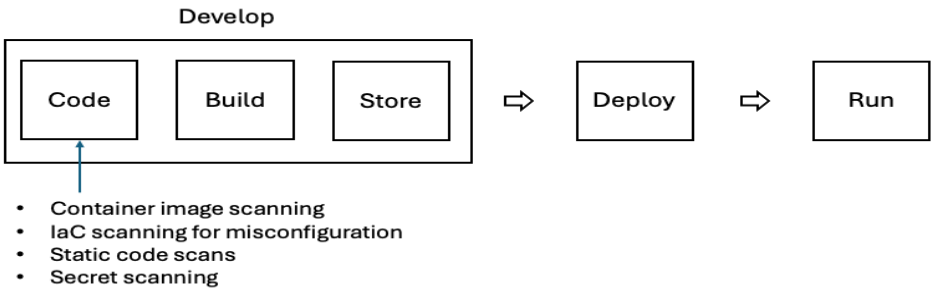

Figure 2 Shift-left security assessment at code stage

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

Various SAST (Static Application Security Testing) tools are available to perform code scans and dependency scanning. Tools like GitHub Advanced Security (GHAS) and SonarQube provide these code and dependency scanning capabilities.

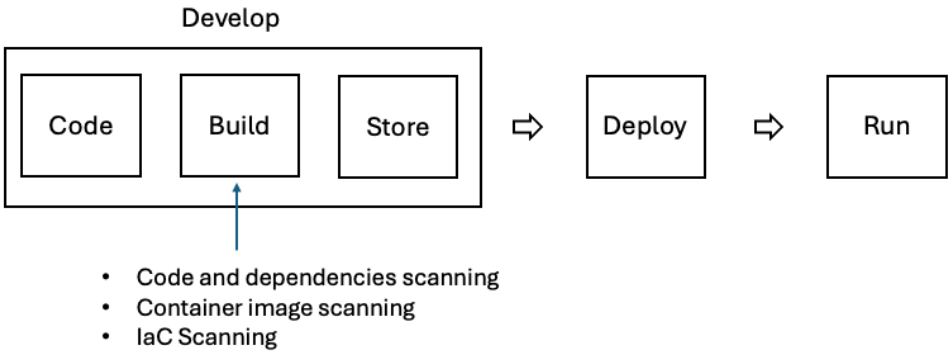

Figure 3 Shift-left security assessment at the build stage

Not all dependencies are available during the static code scan stage. Depending on the library and the framework used for import dependencies, they might be scanned during the coding stage or can be scanned when they are checked into the code repository. During the build time, all required dependencies are resolved, and all layers of containers are downloaded to build the final container image. Vulnerability assessment of the container image and the built artifact will enable us to catch the final artifact that is ready to be deployed. Tools used for performing security assessments can do an assessment of the artifacts produced from this stage.

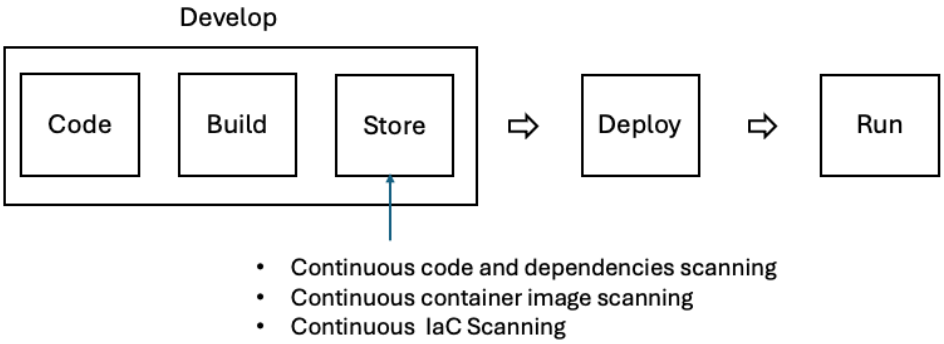

Figure 4 Shift-left security assessment at store stage

Built artifacts are then stored in a central repository. Nexus, Artifactory, Gitlab Package registry, Azure Artifacts, AWS CodeArtifact, harbor, and more tools are available for storing build artifacts. Continuous vulnerability assessment should be performed on these stored artifacts. The main purpose of doing regular scans is to uncover new vulnerabilities that weren't known during the code or build stage. Artifacts like code libraries don't get modified often, but new vulnerabilities are being discovered regularly. Doing a regular scan of stored artifacts is a must to implement a shift-left approach.

The findings from all these substages can be feedback to the user through appropriate channels. Code findings can be pushed as PR to the code repository. Security findings found during build and store substages can be fed back to users via other collaboration tools or published to a central database with dashboards built on top of it.

4.2 Deploy

Security assets at the deployment stage are primarily targeted at the environment-specific configuration, target runtime-specific check, and security policy conformance. At this step, there is a chance to validate the final artifact in junction with the dynamic configuration used for the final deployment. Integrating a checkpoint takes advantage of the last opportunity before security vulnerability or misconfiguration is deployed to production. For container-based

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

deployment, using the container platform-provided controls like Admission Controller in the case of Kubernetes, one can implement the acceptable security posture of the container before it is allowed. Off-the-shelf container security products have their own admission controller implementation, which will give more flexible options for designing and implementing an acceptable security posture for the containers before they get deployed. This is the key stage in the CI/CD pipeline.

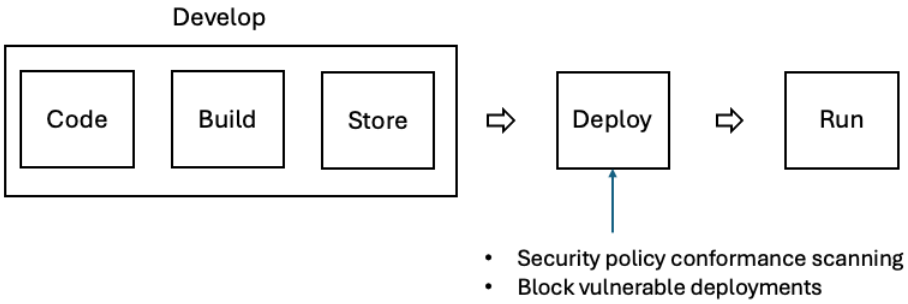

Figure 5 Shift-left security assessment at deploy stage

For organizations looking to shift-left, this would be the last stop before things are pushed to production. For container workloads, container vulnerability posture and configurations should be assessed before deployment. Similarly, IaC code, like terraform scripts, can be validated for known security misconfiguration, and organization-specific policies can be assessed. The security findings from this stage could be fed back to the user via the deployment logs or through other collaboration channels.

4.3 Run

The aim of the Shift-left approach is to minimize the number of security vulnerabilities in production. In the run stage, where the actual application is running, it is essential to monitor the security posture of the running environment. Runtime security monitoring with Endpoint Detection and Response tools like Crowdstrike, Microsoft Defender, SentinelOne, or TrendMicro is a must to have continuous monitoring. Application behavior should be monitored using application logs and custom detection signals for deviations in application behavior like a large number of login attempts or login failure, particularly sensitive functional call behavior deviations. DAST (Dynamic Application Security Tools) can be used at this stage to assess the application security posture further. Generally, DAST tools are not used in the production environment but in pre-production environments, which mimic the production environment. The shift-left approach doesn't eliminate the need for security controls and assessment performed at the final stage of the software development lifecycle; it advocates for doing more during the early stages of the CI/CD with an aim to minimize the number of security findings in a production environment.

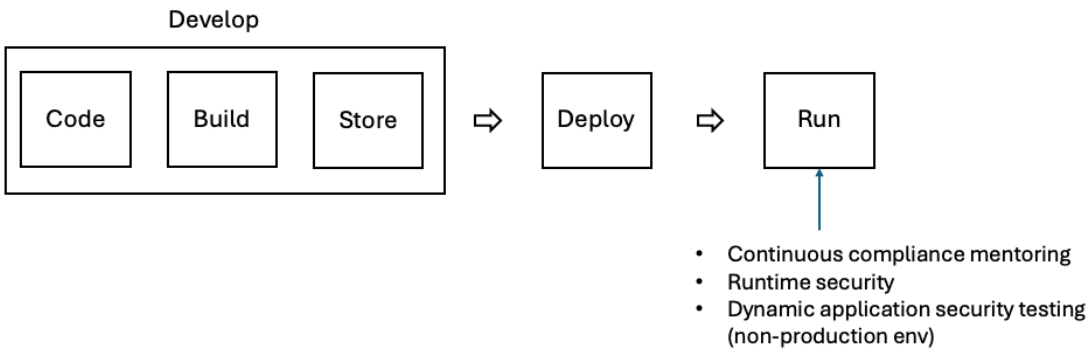

Figure 6 Shift-left security assessment at run stage

Shifting left would mean additional vulnerability assessments generating more findings, and more often, the same findings show up in multiple stages if the code moves through the pipeline without remediating the findings from the

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

initial stages of the CI phase. In practice, application developers are overwhelmed with a number of findings and the redundant nature of the findings. It is important to adopt a vulnerability and security misconfiguration prioritization strategy and apply that to filter the findings and surface the important and impactful findings to the developers to fix it.

V. CHALLENGES WITH SHIFTING-LEFT

In reality, there are significant challenges before adopting a shift-left security approach. Many organizations are yet to embrace it fully; there are about only 37% of the organization is fortune 500 have integrated security into their DevOps processes extensively []. Some of the key challenges are**.** Organization Culture: Engineering teams are often measured by productivity metrics like pull requests or features delivered, whereas shift-left security requires a focus on vulnerability prevention and early remediation; this would be a cultural shift and new performance metrics. Security first culture and culture rewarding lesser security findings in a production environment would encourage organizations to shift left. Tool Silos: Security teams and development teams use vastly different tools resulting in poor visibility across tools and platforms. Developers' chosen tools often excel in aiding them with running the operations of the platform or application and often lack insight into security risks, while security teams' tools struggle to provide the capabilities needed by the operations teams. Skillset Gap: For the shift-left approach to be effective, you would need Information Security teams to have deep knowledge about the platform and everyday operations and device process that doesn't hamper the operation teams' workflows or impact their ability to provide great service to their customer. On the other hand, IT platform operators often don't have the security depth and sometimes motivation to implement strict security controls. When faced with situations where they must choose between agility and security, operations teams often favor agility. The lack of established processes and collaboration between InfoSec and engineering teams hinders cross-functional efforts to implement security from the beginning of development. Alert Fatigue: With security assessment at every stage of the CI pipeline, there is a steady flow of alerts and, most often, duplicate alerts. The same vulnerability or misconfiguration is being flagged at various stages of the CI pipeline. The flood of alerts and lack of proper context and correlation will lead to alert fatigue. This, coupled with the complexity of managing multiple tools, will be an additional burden for application owners.

VI. CONCLUSION

This paper has provided a detailed, stage-by-stage approach to implementing shift-left security within the continuous integration and delivery (CI/CD) pipeline. By breaking down the security measures to be taken at each phase, develop, build, store, deploy, and run, the paper offers a clear roadmap for organizations aiming to integrate security into their software development lifecycle. Each stage was explored in depth, illustrating the tools and techniques available to detect vulnerabilities early and mitigate risks before they propagate. This approach serves as a practical blueprint for development and security teams, enabling them to embed security checks seamlessly into existing workflows. By following this approach, organizations can build upon the shift-left methodology, adapting it to their specific needs and tools while ensuring that security is addressed at the earliest and most critical points in the development process. The guidelines presented here can form the foundation for a robust, scalable security strategy that evolves with technological advancements and new threat landscapes.

REFERENCES

1. Checkpoint. (2020). What is shift left security? Checkpoint. https://www.checkpoint.com/cyber-hub/cloud-security/what-is-shift-left-security/
2. Aqua Security. (2021). Shift left in DevOps. Aqua Security. https://www.aquasec.com/cloud-native-academy/devsecops/shift-left-devops/
3. Fortinet. (2022). Shift left security. Fortinet. https://www.fortinet.com/resources/cyberglossary/shift-left-security
4. GitGuardian, What is shift left security. https://www.gitguardian.com/glossary/what-is-shift-left-security
5. JFrog, What, why, and how of shift left security. https://jfrog.com/devops-tools/article/what-why-how-of-shift-left-security/
6. SparkFabrik. Container security: How to. https://blog.sparkfabrik.com/en/container-security-how-to
7. GitLab. CI/CD. https://about.gitlab.com/topics/ci-cd/
8. GitHub. Application security testing. https://github.com/resources/articles/security/application-security-testing
9. Palo Alto Networks. (2023). Shift left security. Palo Alto Networks. https://www.paloaltonetworks.com/cybersecurity-perspectives/shift-left-security
10. Wiz. Shift left security. https://www.wiz.io/academy/shift-left-security
11. Wiz. Wiz magic shifts left. https://www.wiz.io/blog/wiz-magic-shifts-left

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 10, October 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1110001 |

12. APISec. (2022). Shift left security. APISec. https://www.apisec.ai/blog/shift-left-security
13. Snyk. (2021). Shift left security. Snyk. https://snyk.io/learn/shift-left-security/
14. Google Cloud. (n.d.). Shift left on Google Cloud security: Invest now, save later. Google Cloud. https://cloud.google.com/blog/products/identity-security/shift-left-on-google-cloud-security-invest-now-save-later
15. ReversingLabs. (2023). Why "shift left" is now a dirty word in some security circles. ReversingLabs. https://www.reversinglabs.com/blog/why-shift-left-is-now-a-dirty-word-in-some-security-circles
16. Orca Security. (2022). Shift left security platform. Orca Security. https://orca.security/resources/blog/shift-left-security-platform/
17. Li, Y., Chen, X., & Zhao, J. (2021). Shift-left testing in DevOps: Challenges and solutions. Proceedings of the 4th International Conference on Software Engineering and Applications. https://doi.org/10.1145/3475716.3475786
18. Hussein, A. M. (2022). Shift left approach in software security. Baghdad Journal of Networks, 5(2), 123–130. https://mesopotamian.press/journals/index.php/BJN/article/view/488/355
19. CrowdStrike. Shift left security. https://www.crowdstrike.com/cybersecurity-101/shift-left-security/
20. Palo Alto Networks. (2019, July 9). 4 practical steps to shift left security. Palo Alto Networks. https://www.paloaltonetworks.com/blog/2019/07/4-practical-steps-shift-left-security/

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details