Transitioning from DevOps to DevSecOps

Sebastian Sjövald, sjovald@kth.se
Vilma Jalava, vilmaj@kth.se

April 2022

1 Introduction

DevOps is altering the way software companies develop, deploy, and service their products by being able to increase their frequency of deployments by automating them. Many more companies want to use DevOps, but they are concerned about the security of the software they develop. As a result the practice of DevSecOps has has grown. This alludes to the integration of security principles into a DevOps environment by encouraging collaboration between development, operations, and security teams. However this is a farily new concept which entails a lack of experienced personnel, research and tools. In this essay we will look at the two concepts, DevOps and DevSecOps, transitioning between them aswell as comparing common DevSecOps tools that can be used.

2 Background

2.1 DevOps

DevOps is a set of practices that combines development (devs) and operations (ops) to increase the efficiency of software development and deliveries. [1] Before DevOps, two different teams worked apart from one another, one team of developers who wrote code worked and an operations team who deployed and supported the code. This resulted in a slow process of releasing new features or updates. [2]

DevOps consists of eight phases, divided into four main categories, as exhibited in Figure 1 [3]. The first category is Continuous Integration which consists of the Code and Build phases of the DevOps pipeline. [1] This is the practice of developers regularly merging their code changes into a central repository, after which automated builds and tests are run. By having continuous smaller merges, we are more likely to find more minor bugs and avoid the scenario of making massive merges infested with bugs.[4]

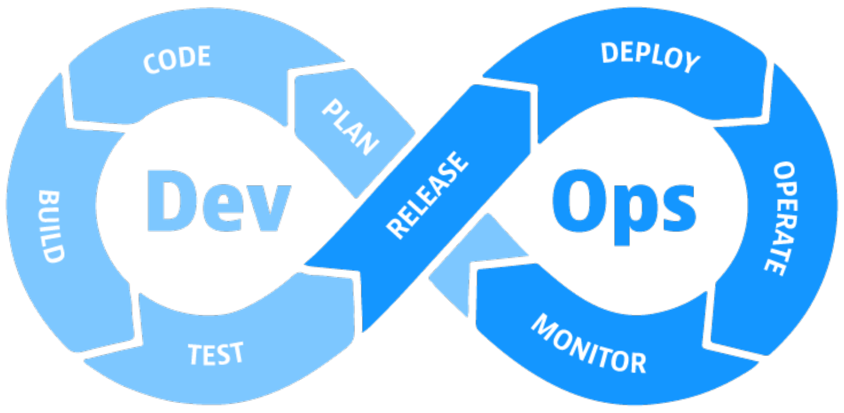

Figure 1: The eight DevOps phases

Continuous delivery is a Continuous Integration add-on that automates the process of deploying a new build to production. Continuous delivery aims to: Perform automated testing on each new build to verify that it is ready for production and fail those that aren't. It also aids in managing deployment environments' automatic provisioning and configuration and their testing for stability, performance, and security compliance. When a new release is approved and manually triggered by the organization, it is deployed into production. [5]

Continuous Delivery links with the pipeline's Test and Release phases, allowing companies to manually trigger the release of new builds as often as they want. Continuous Deployment is a more advanced variant of Continuous Delivery where the manual step of approving new releases for production is no longer necessary.[6] Each build that passes all of the pipeline's checks and balances is automatically deployed into production under a Continuous Deployment paradigm.

Continuous Feedback links the loop's ends together, feeding data and analytics from the Operate and Monitor phases back into the Plan phase to repeat the process. [7] The whole purpose of DevOps is to get new updates out as quickly and efficiently as possible so that the company can gather input for its next planning phase.

2.2 DevSecOps

Because DevOps enables developers to deliver and deploy updates and modifications rapidly, the security process must follow suit to maintain the product's security criteria. This becomes difficult for an isolated security team unless their work is integrated into the DevOps process. Traditionally, the security team examined the code near the end of development, just before deployment. When development cycles are shorter, developers must incorporate security into daily activities. In regular DevOps, the security aspect was only in the testing phase, but in DevSecOps, you build a secure foundation for DevOps initiatives. [8] A big difference is that when in DevOps, the security is only being tested at the end phase; in this implementation, it is tested throughout the deployment, in every step, referred to as "shifting left." If we envision the phases as a horizontal timeline, shifting something left indicates it gets completed sooner in the timeline, resulting in any program security issues being discovered earlier in the development process.[9] However, the end security tests used in DevOps are still in place in the final stage.

3 The transition from DevOps to DevSecOps

The first big step to be taken when switching from DevOps to DevSecOps is a cultural transition. Instead of having some people responsible for the deployment and others for security testing, every single person working on the product should consider the security aspect of what they do. There are many tools available to help with security management within continuous deployment, but ensuring that they are used effectively entails instilling the belief that everyone is accountable for security and fostering a trusting environment where everyone collaborates. [10]

Secondly, the company needs to instil procedures for the different phases of security. The initial step is prevention. This comprises both preventative measures and a strategy for minimizing the consequences were something to happen. Another component is detection and analysis when the company actively searches for and analyzes potential dangers. This is similar to the traditional DevOps testing phase; however, it is now performed in every phase. Containment and recovery is the third procedure, a plan for recovering from an incident and isolating damaged elements from the rest of the technology in case of a security breach. [11] [12]

3.1 Challenges

One of the most significant transformations that need to be done when switching from DevOps to DevSecOps is within the culture. This shift entails instilling the belief that everyone is accountable for security and fostering a collaborative environment where everyone strives to improve security measurements. It always takes time to introduce significant changes to people's working methods since most employees already have their habits and routines. The organisation must develop new collaborations between developers and security specialists to implement sharing. This necessitates the formation of new teams. Traditionally, security specialists and developers have worked in opposition, slowing each other down. Developers write not-secure code, which security specialists must correct. However, bridging these two parts of a software development organisation is critical for the DevSecOps aspect of exchanging data, knowledge, and education.

Another challenge in implementing DevSecOps is the lack of expertise, tools, and solutions. These issues are caused by DevSecOps being a relatively new concept that has yet to become an industry standard. [13]This is partnered with developers getting new responsibilities which they might not be used to, which can lead to increased pressure for the developers. [14]

3.2 Adopting DevSecOps tools

One of the core principles in DevSecOps is automation and keeping security an aspect during every part of the way in the software development lifecycle. A prominent way of doing this is by involving security into development in the form of tools. DevSecOps tools help developers gain insight into the security aspect of their code and prevent possible attacks from happening. Depending on your current DevOps environment integrating one or more of these could help your organization shift left on security.

3.2.1 Static Application Security Testing

Static Application Security Testing[15] which is abbreviated as SAST is the practice of staticly analyzing an application. As it doesn't require running code it can be performed very early in the software development life cycle. SAST helps identify possible security risks and in certain cases highlight the code that is prone to security faults. This type of approach is called white-box testing as the tool has access to the entire codebase when analyzing. Automation is a must in a DevOps environment as well as in a DevSecOps one. SAST can be implemented so that on each push to a repository the tool is executed and developers can get real-time feedback regarding the security of their application.

Open-source implementations that can be used on your own projects are:

- NodeJsScan
- SonarQube
- Bandit
- CodeSonar

But many more are available and you can look for your own depending on which framework or programming language you are using.

3.2.2 Dynamic Application Security Testing

Dynamic Application Security Testing[16] is in contrast to SAST performed when the application is running. This is another automation approach where the application is testing by a tool instead of a human being for known security issues. Enought though this approach comes later in the software development life cycle it can discover issues such as run-time or environment-related issues. An approach such as DAST represents a hackers point of view of the application. It sees the application as a black box and performs known attacks against vulnerbilities that are common or known in other ones.

Some open-source implementations that can be used on your own projects are:

- ZAP
- Nikto2
- GoLismero

In the same manner as the SAST tools, the DAST tool needs to be investigated in regards to what is relevant to your application. DAST tools are not equally framework-or language-dependant but rather on what it tests. Do you want to test for vulnerablities such as SQL-injections then look for a tool that supports that option.

3.2.3 Interactive Application Security Testing

Interactive Application Security Testing[17] is a hybrid way of analyzing security faults in an application. Unlike SAST and DAST, IAST works by placing an agent in the application that can analyze its faults in real-time. This combines the positive functionalities of SAST and DAST by giving more detailed information if errors are found. For example if security fault is found during run-time of an application, a normal DAST cannot say where the fault is located in the source code only which page that has it. A SAST would not be able to find the fault as it is only caused during run-time. By using IAST, the agent has access to both parts and would be able to show where the fault was located in the source code. However by inserting the agent directly into the application can cause performance impacts. This might be unwanted in production but as you want to "shift left" on security in a DevSecOps environment you might want to consider enabling the agent during development.

IAST is as SAST language or framework-dependant. If you are considering integrating a IAST into your DevSecOps environment you need to consider both the performance impact and the amount of vulnerabilities that that specific IAST will be able to detect before you decide if it is relevant to add it.

3.2.4 Run-time Application Security Protection

Run-time Application Security Protection[18] works in a similar way to the IAST tool where an agent is placed inside the application and can observe both run-time issues as well as the source code. The main difference between IAST and RASP is the way they handle security issues. RASP is a protection tool that, in the case of a possible attacks, defends the application instead of analyzing and testing it for further development. Looking at the software development life cycle RASP is used to maintain security after the application is released into production. However, information about attacks and what defenses has been triggered can help further the development of the security side of the application.

RASPs are not language or framework-dependant but as IAST, RASP can have an performance impact on the application which can be a factor to consider when choosing whether to integrate it or not.

4 Conclusion

In order to move from having a DevOps environment to a DevSecOps one, security is a must in every part of the way during development. This can be a cultural challenge for many teams but having transparency helps negate issues such as security teams having to correct common vulnerabilities after code has been pushed to production. There are many tools for shifting left on security. While some covers more "ground" such as IAST by having access to both run-time and source code, your environment may not be suitable for integrating it. By having too many automation tools it can lead to longer test times which in its turn leads to longer feedback cycles. This goes against the core principles of DevOps and without DevOps you can't have DevSecOps. So reflect on your organizations current structure, look into what alternatives and tools exist, then try to integrate security into the whole development cycle. Only by doing that can you transition from having a DevOps environment to having a DevSecOps one.

References

[1] Liming Zhu, Len Bass, and George Champlin-Scharff. “DevOps and Its Practices”. In: *IEEE Software* 33.3 (2016), pp. 32–34. DOI: 10.1109/MS.2016.81.

[2] *DevOps principles.* Accessed 1 May, 2022. URL: https://www.atlassian.com/devops.

[3] Ramtin Jabbari et al. “What is DevOps?: A Systematic Mapping Study on Definitions and Practices”. In: May 2016, pp. 1–11. DOI: 10.1145/2962695.2962707.

[4] *what is devops? - amazon web services.* Accessed 1 May, 2022. URL: https://aws.amazon.com/devops/what-is-devops/#:~:text=DevOps%5C%20is%5C%20the%5C%20combination%5C%20of,development%5C%20and%5C%20infrastructure%5C%20management%5C%20processes.

[5] Floris Erich, Chintan Amrit, and Maya Daneva. “A Mapping Study on Cooperation between Information System Development and Operations”. In: vol. 8892. Dec. 2014, pp. 277–280. ISBN: 978-3-319-13834-3. DOI: 10.1007/978-3-319-13835-0_21.

[6] Floris Erich, Chintan Amrit, and Maya Daneva. “A Qualitative Study of DevOps Usage in Practice”. In: *Journal of Software: Evolution and Process* 00 (June 2017). DOI: 10.1002/smr.1885.

[7] Miguel A. López-Peña et al. “DevOps for IoT Systems: Fast and Continuous Monitoring Feedback of System Availability”. In: *IEEE Internet of Things Journal* 7.10 (2020), pp. 10695–10707. DOI: 10.1109/JIOT.2020.3012763.

[8] Håvard Myrbakken and Ricardo Colomo-Palacios. “DevSecOps: A Multivocal Literature Review”. In: *Software Process Improvement and Capability Determination.* Ed. by Antonia Mas et al. Cham: Springer International Publishing, 2017, pp. 17–29. ISBN: 978-3-319-67383-7.

[9] Hewlett Packard Enterprise. *Application security and devops.* Tech. rep. Technical report, Hewlett Packard Enterprise, 2016.

[10] Akond Ashfaque Ur Rahman and Laurie Williams. “Security Practices in DevOps”. In: *Proceedings of the Symposium and Bootcamp on the Science of Security.* HotSos ’16. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2016, pp. 109–111. ISBN: 9781450342773. DOI: 10.1145/2898375.2898383. URL: https://doi.org/10.1145/2898375.2898383.

[11] Vaishnavi Mohan and Lotfi Ben Othmane. “SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps”. In: *2016 11th International Conference on Availability, Reliability and Security (ARES).* 2016, pp. 542–547. DOI: 10.1109/ARES.2016.92.

[12] Fabiola Moyón et al. "Integration of Security Standards in DevOps Pipelines: An Industry Case Study". In: *Product-Focused Software Process Improvement*. Ed. by Maurizio Morisio, Marco Torchiano, and Andreas Jedlitschka. Cham: Springer International Publishing, 2020, pp. 434–452. ISBN: 978-3-030-64148-1.

[13] Roshan Namal Rajapakse, Mansooreh Zahedi, and Muhammad Ali Babar. "An Empirical Analysis of Practitioners' Perspectives on Security Tool Integration into DevOps". In: *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. New York, NY, USA: Association for Computing Machinery, 2021. ISBN: 9781450386654. URL: https://doi.org/10.1145/3475716.3475776.

[14] Gerry Gerard Claps, Richard Berntsson Svensson, and Aybüke Aurum. "On the journey to continuous deployment: Technical and social challenges along the way". In: *Information and Software Technology* 57 (2015), pp. 21–31. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2014.07.009. URL: https://www.sciencedirect.com/science/article/pii/S0950584914001694.

[15] *Static Application Security Testing*. Accessed 2 May, 2022. URL: https://www.synopsys.com/glossary/what-is-sast.html.

[16] *Dynamic Application Security Testing (DAST)*. Accessed 2 May, 2022. URL: https://www.rapid7.com/fundamentals/dast/.

[17] *Interactive Application Security Testing (IAST)*. Accessed 2 May, 2022. URL: https://www.synopsys.com/glossary/what-is-iast.html.

[18] *RASP SECURITY*. Accessed 2 May, 2022. URL: https://www.contrastsecurity.com/knowledge-hub/glossary/rasp-security.