DevOps and Security - How to fit them together?

Yilin Chang (yilinc@kth.se)
Zehao Jiang (zehaoj@kth.se)

April 5, 2022

1 Introduction

DevOps is a software development practice that facilitates collaboration between development (Dev) and operations (Ops) to deliver software faster and more reliably. However, in order to achieve its fast path, certain things are left out and one of them is security. The nature of DevOps brings security risks to the development process, which could be costly to fix. To address this issue, the concept of DevSecOps was introduced. By integrating security across the lifecycle of the software, it will greatly reduce the cost of fixing errors and bugs and allow the team to deliver more secure software at high speed. But this concept still remains overlooked compared to the traditional one, and one of the reasons is that DevSecOps is a relatively new idea and is hard to achieve. So, how could we actually fit security into DevOps?

In this essay, we will illustrate how DevOps and security could fit together. Integrating security into DevOps can be done from both team and tools aspects. All developers must keep the idea of "shift right" and putting security in all phases in mind. Besides, there are multiple DevSecOps tools that can be helpful in different areas and stages, including access control, penetration testing and static code review. Based on all this, we will discuss and share our thoughts on the future trend of DevSecOps.

2 DevOps VS DevSecOps

2.1 Traditional DevOps and its security risks

A software development life cycle or SDLC usually consists of the following phases: planning, coding, building, testing, releasing, deploying, operating, and maintaining(monitoring). Initially, the program was relatively simple and the programmer alone could complete all phases of the work. As the software industry grew and grew, the size of the software became progressively larger and the complexity of the software kept climbing. Software developers spend weeks writing code; then the code is handed over to the Quality Assurance team for testing; then the final release is handed over to the operations & maintenance team for deployment. This early software delivery model was called the ”Waterfall model”.

This model is suitable for projects with more idealized conditions, very clear user requirements, and very good development time. However, projects cannot work in one direction as customers have needs too and products have problems and need to be improved. Thus the famous ”Agile Development” was introduced and DevOps is an evolution of such movement. Started in 2008 with developers Andrew Clay and Patrick Debois, it aims to overcome the commonly seen problems in agile development such as decreased collaboration over the size of the project, and the negative impacts of incremental delivery on long-term outcomes[1].

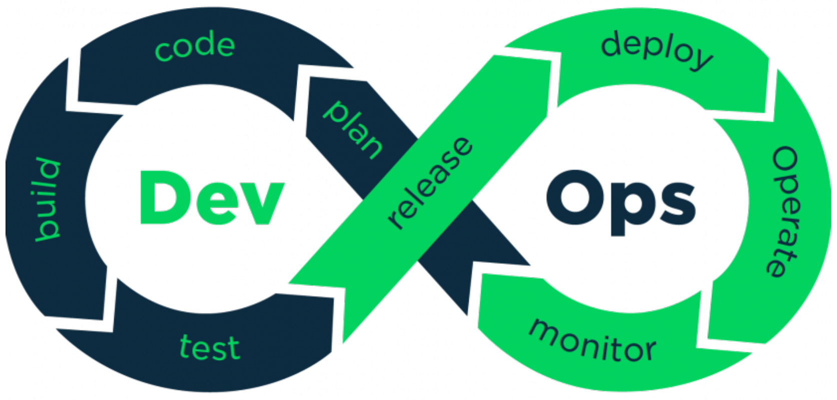

Figure 1: DevOps workflow[2]

The workflow of DevOps can be represented in *Figure 1*. DevOps aims at removing the barrier between the development (Dev) and operations (Ops) teams across the software development lifecycle. DevOps teams collaborate to continuously build, release and manage software in faster, more frequent cycles. By assuring faster deployment, it can not only reduce the risk of having major mistakes but also increase the quality of the product by constantly having nearly real-time user feedback. The close connection between different teams can at the same time allow easier communication and boost efficiency.

However, The shift from a traditional IT model to the newer DevOps brings additional security challenges to the team. DevOps is not perfect and has been criticized for its lack of consideration for security. Actually, most of its security challenges are caused by the advantages.

Developers aim to move their software through the pipeline as quickly as possible, while security teams focus on bugs that could hinder development, thus arise the conflicts. By nature, security teams tend to spend longer time looking through every part of the code. Security is often sacrificed for the sake of speed, thus allowing a great number of risks like misconfigurations and unresolved flaws to stay alive in the software. The fast path in the meantime can lead to an increasing number of coding mistakes, which result in bugs and errors that attackers could exploit.

One of the most common yet serious security threats in DevOps is identity and data security. The cloud-first architecture of DevOps brings a much broader attack surface. When conducting a migration of data or applications to the cloud, the exposure of sensitive data is a big concern. Since DevOps requires different teams to collaborate and interact constantly, developers and operators keep sharing critical data, SSH keys, APIs, tokens, and other information in order to keep up with the fast speed. As a result, these confidential assets will be passed through a lot of platforms and become leaking points.

2.2 DevSecOps

With all the risks mentioned above and the fact that DevOps neglects security by its fast-speed nature, a concept called DevSecOps was introduced. DevSecOps (development security operations) is an organizational model dedicated to incorporating security, including scanning, monitoring, and remediation, into DevOps. It aims to apply security across the SDLC, which encompasses all phases in DevOps-from planning, developing, building, and testing all the way to release, deployment, operations, and updates. This helps reduce the cost of security and allows the team to deliver secure software more quickly.

The biggest difference between DevSecOps and traditional security operations is that DevSecOps introduces security activities early in the SDLC, instead of waiting until the product is released. This is also one of the key concepts in DevSecOps–"shifting left". "Shifting left" means the team should find defects and guarantee the product's security at the earliest stages possible in the development workflow. This practice can reduce both time and costs in development compared to traditional security testing. As traditional testing is only implemented during the last phases of the development lifecycle, the bugs may have existed for a long time before being discovered. The developers then need to spend a long time reviewing the old code, fixing bugs and relevant dependencies, which in the end increases developing costs. Shortening the time gap between committing and receiving a security warning, which "shifting left" intends to do, will enable developers to fix bugs more quickly and avoid unnecessary changes. *Table 1* shows relative cost to repair defects when found at different stages of software. The benefit would be huge if we can detect products defects in early stages.

3 Integrating security to DevOps

To integrate security into DevOps, the team needs to make changes from traditional security measures to DevSecOps in many aspects: from team structure,

Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	**5X**	**10X**	**15X**	**30X**

* X is a normalized unit of cost and can be expressed terms of person-hours, dollars, etc.

Table 1: Relative Cost to Repair Defects When Found at Different Stages of Software Development[3]

and culture to security practices and tools. Every DevSecOps project is unique and there are many practices to follow to perform DevSecOps. In this section, we will discuss how to do it from two aspects: team and tools.

3.1 DevOps team

The human factor is always the weakest link in a system, which is the same case for DevSecOps. No matter how many technologies and tools were introduced to improve security in DevOps, it would be useless if people are reluctant to use them. In traditional software development, the security team is separated from others and is the only one responsible for the security of the final product. However, to implement DevSecOps, security in all DevOps phases is needed and all developers are required to participate in security testing.

Therefore it's important to educate developers about application security, the modern threat landscape, and security best practices for the specific programming languages and systems they work on. It's also essential to form a culture in the company–where everyone in different teams is ready to communicate and help each other and everyone is responsible for the security of the product. These can be achieved by setting up security champions instead of security team[4]. Security champions are members that experts in the security of software development. They can teach and help other developers about all the security issues during development, emphasize security concerns in the whole team, help with QA and testing, and so on.

3.2 DevSecOps Tools

Tools are an essential part of DevSecOps. As in the fast-paced DevOps environment, it's inefficient and impossible to follow the DevSecOps practices and secure DevOps manually. Security needs to be automated and integrated with the automation pipelines as well, such as Continuous Integration (CI) and Continuous Delivery (CD). These processes require a large number of tools and information systems[5].

DevSecOps is a new paradigm and does not yet have an established toolset. Big organizations like Netflix, Google, and Amazon have their own tools, while most others use existing ones. There are many DevSecOps tools today in the market, and they can roughly be classified into two groups[6]. The first group aims to reduce security risks in DevOps pipelines, detecting and fixing security vulnerabilities through continuous security testing. The second one aims to support security teams, providing them with automated security reports so that they don't need to manually review and approve every release of the project. More specifically, each tool is dedicated to enhancing DevOps security from one or several perspectives and is applied in different stages. In the remaining of this section, we will briefly introduce three popular DevSecOps tools: HashiCorp Vault, OWASP ZAP, and SonarQube.

3.2.1 HashiCorp Vault

HashiCorp Vault is a tool that manages and secures access to secrets. The secret can be anything we want to control access to, such as API encryption keys, passwords, or certificates[7]. Vault provides "encryption as a service", encrypting and decrypting data without storing it[8]. This protects your secret data from the threat of unauthorized access during transitions. It can also generate dynamic secrets on-demand for different systems, such as AWS or SQL databases.

Vault also provides Identity-based access control. It authenticates users either via passwords or temporary tokens which allows you to access a particular path. Then it grants different access rights to the user based on the authenticated identity and records the access history of the user.

All these features of HashiCorp Vault make it an excellent tool for teams to manage secrets in a DevOps environment, enabling the team to share secrets both conveniently and securely.

3.2.2 OWASP Zed Attack Proxy (ZAP)

OWASP ZAP is a free, open-source penetration testing tool designed to help developers practice better software security. Penetration testing(pentesting) is a way of testing that testers act as an external attacker to break into the system, which helps uncover the underlying vulnerabilities in the system.

ZAP works as a "man in the middle proxy". It stands between the tester's browser and the web application so that it can intercept, inspect, modify and forward the messages sent between the browser and web application[9]. Besides finding security vulnerabilities manually, ZAP also provides automated scanners which you can integrate into your CI/CD pipeline and perform pentesting automatically.

3.2.3 SonarQube

SonarQube is an open-source tool that can automatically perform code reviews in more than 20 different languages. It can detect bugs, code smells and vulnerabilities in your code. By integrating into the team's workflow, it can provide continuous code review across all project branches and pull requests.

SonarQube provides two kinds of code review based on the severity of the security issue in the code: security hotspots and vulnerabilities[10]. Hotspots are uses of security-sensitive code, which may not contain security risks but still need further inspection to be sure. Vulnerabilities, on the other hand, need immediate actions. Additionally, SonarQube also provides security reports, including security overviews and the top security risks of the project.

4 Discussion - Future trends of DevSecOps

As more and more people adopt the concept of DevSecOps, it will no doubt continue growing and advancing in the coming years. Though overlooked compared to the traditional DevOps, we cannot neglect the fact that it's drawing more attention and will continue doing so in the future, as seen in the google trend in *Figure* 2 as well as the number of search results in Google Scholar using the term *DevSecOps* in *Table* 2.

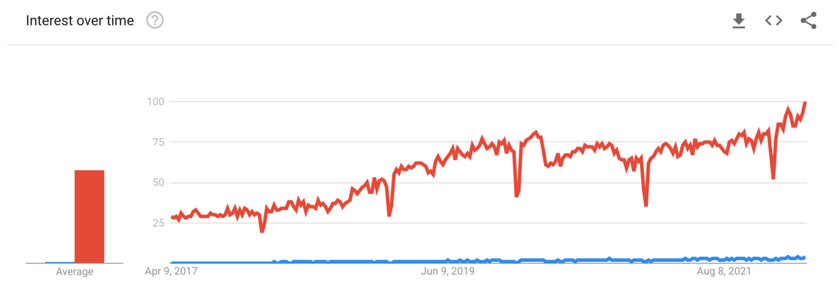

Figure 2: Google trend for term *DevOps* and *DevSecOps* in the last five years

Year	Number
2017	50
2018	100
2019	230
2020	411
2021	544
2022 (first quarter only)	125

Table 2: Number of google scholar search results with term DevSecOps

Organization: DevSecOps engineers are now focusing on security and risk prevention protocols in development workflows to make sure that risks are found as soon as possible in the whole process. In the future, we believe that there will be more security and compliance controls getting integrated at even earlier stages into SDLC. The protection throughout the data lifecycle and access control will be seen more frequently across all aspects of the digital transformation.

Regulations or tools to deal with data, access, or identity leakage will also be introduced.

Tools and Automation: According to Shackleford et al.[11], there is a lack of available tools. At the same time, with more companies adopting DevSecOps, more hands are needed and the answer to that is automating workflows, testing, and development. Great opportunities lie in the field of AI-driven applications to improve security, with AI becoming more common in areas like threat detection, threat response, and threat auto-correction. Great amounts of time will be saved and both efficiency and performance will be increased.

5 Conclusions

The introduction of DevOps model makes the development more flexible and less time-consuming, while the security process gradually become a bottleneck. The advent of DevSecOps solved this problem, by integrating security into each of DevOps phases.

The key concept of DevSecOps is ”shifting left” - to perform security testing early in the development lifecycle. This helps reduce the delay and cost of security tasks. Furthermore, two points are needed to keep in mind when implementing DevSecOps. First, every member in the team should have knowledge and awareness of the security of the project. We can set up security champions to help achieve this. Second, use suitable tools to help you integrate security into different phases of DevOps pipeline. Examples include HashiCorp Vault for security management, OWASP ZAP for penetration testing, and SonarQube for static application security testing (SAST).

DevSecOps, a new concept in software technology, is not very popular at this moment but is receiving more and more attention every year. An increasing number of companies are adopting DevSecOps, and more research and tools are emerging in the field. In particular, it is considered to have considerable potential when it comes to using artificial intelligence in this area.

References

[1] S. Gunja, “What is devops? unpacking the rise of an it cultural revolution.” `https://www.dynatrace.com/news/blog/what-is-devops/`.

[2] “Must-know technologies for devops engineer in 2021.” `https://solomotechnology.com/must-know-technologies-for-devops-engineer-in-2021/`, 2021.

[3] S. Planning, “The economic impacts of inadequate infrastructure for software testing,” *National Institute of Standards and Technology*, p. 1, 2002.

[4] M. Maxwell, “20 devsecops best practices across people, process and technology.” `https://www.contino.io/insights/devsecops-best-practices`, 2020.

[5] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “Devops,” *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.

[6] Aqua, “Devsecops tools: 9 ways to integrate security into the sdlc.” https://www.aquasec.com/cloud-native-academy/devsecops/devsecops-tools/.

[7] H. Vault, “What is vault?.” https://www.vaultproject.io/docs/what-is-vault.

[8] A. Lugger, “What is hashicorp vault and how does it work?.” https://sensu.io/blog/what-is-hashicorp-vault-and-how-does-it-work, 2020.

[9] O. ZAP, “Getting started.” https://www.zaproxy.org/getting-started/.

[10] SunarQube, “Code security, for developers.” https://www.sonarqube.org/features/security/.

[11] D. Shackleford, “A devsecops playbook,” *SANS Institute*, 2016.