DevOps and Security - How to fit them together?

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

May 30, 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*, which is also often referred to as *Secure DevOps* or *SecDevOps*, was introduced. DevSecOps is the concept of integrating security practices across the lifecycle of the software, instead of only incorporating security tests in the last phase of the pipeline as an extension. Since bugs can be discovered in an earlier phase of development by such practice, it will greatly reduce the cost of fixing them compared to spending a long time reviewing old codes. Shortening the time gap between committing and receiving a security warning also allows 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 by presenting multiple DevSecOps tools that can be helpful in different areas and stages, including access control, penetration testing and static code review. Based on this, we will examine:

- DevOps' security risks and what is DevSecOps
- How to integrate security into DevOps from multiple aspects
- 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). With the expansion of the software industry, the size of the software became progressively larger and emerged finer division of responsibility. The rather strict separation between teams in the traditional software development model causes friction and slows down the whole process of development. Thus, to make things work as agile as possible and eliminate the gaps, 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].

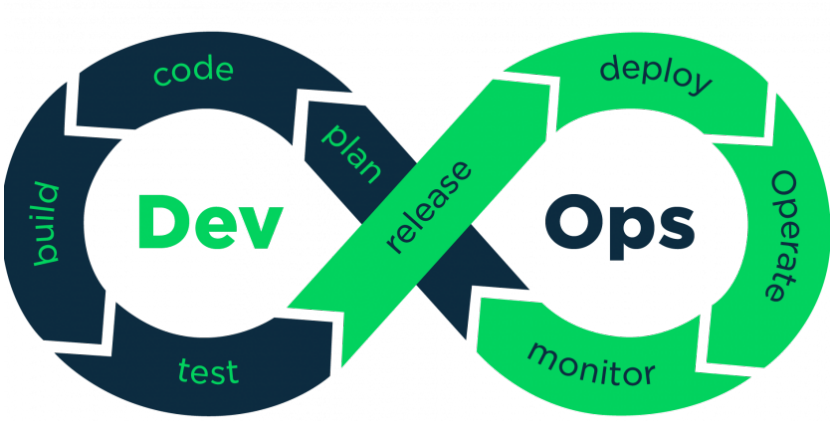

Figure 1: DevOps workflow[2]

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[3].

The different goals of developer teams and security teams causes friction

in between. Developers aim to move their software through the pipeline as quickly as possible, while by nature, security teams tend to spend longer time looking through every part of the code and focus on the stability of the software. Sacrificing security for faster development pace not only increases the number of coding mistakes by developers, but also allows a great number of risks like misconfigurations and unresolved flaws to stay alive in the software.

Another common yet serious security threats in DevOps is identity and data security[4]. 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 a sample costs from Google of bugs found in different phases of testing, and it shows the drastically increasing costs of fixing bugs at later stages in SDLC. The benefit would be huge if we can detect products defects in early stages.

Software Testing Phase Where Bugs Were Found	**Estimated Cost per Bug**
System Testing	$5,000
Integration Testing	$500
Full Build	$50
Unit Testing/Test-Driven Development	$5

Table 1: The cost to fix bugs at Google [5]

3 Integrating security into DevOps

Applying DevSecOps is a systematic process. Many researches on DevSecOps suggests that four aspects should be considered when integrating security: *culture*, *automation*, *measurement*, and *sharing* (CAMS) [6] [7] [8]. In this section, we will discuss how to integrate security into DevOps from these four aspects.

3.1 Culture

Culture, as the first pillar of CAMS, means creating a company culture where security is considered everyone's responsibility. It is an essential yet easily neglected step towards DevSecOps. As shown in the study conducted by Nora Tomas et al., almost all the subjects they interviewed mentioned some form of negligence of security in the routine operations of their companies, such as "Nobody is responsible for security", and "management does not prioritize security" [6].

Mary Sánchez-Gordón and Ricardo Colomo-Palacios conducted a literature review on DevSecOps and identified 13 aspects to establish DevSecOps cultures [7]. Among them, collaboration and sharing knowledge are the two most frequently brought up aspects. Collaboration means integrating security principles in DevOps through more frequent collaborations among development, operation, and security teams, and requires the security team to participate in the whole process of development. Sharing knowledge requires educating all members of every team and increasing their awareness about security, as it is often neglected by non-security members. This can be achieved by setting up security champions [9]. Security champions are members that expertize in the security of software development. They can teach and help other developers about all the security issues during development, emphasize security concerns, help with QA and testing, and so on.

3.2 Automation tools

Automation involves using tools to automate security processes in DevOps. It's also an essential part of the fast-paced DevOps environment. It makes security processes fast, scalable and effective thus making it possible to keep a high pace for detecting, alerting, resolving errors, and finding countermeasures for future

errors [10]. It also makes security more probable to fit into DevOps pipeline, and frees up developers' time to manually set up security tests.

There are many DevSecOps tools in the market today. However, as a new area, there has not yet appeared an integrated solution or toolkit for DevSecOps, and many tools are only specialized in resolving certain security risks. The DevSecOps Fundamentals Guidebook [11] provides a guideline of applying security tools in DevOps, shown as *Table* 2. It shows the security activities and types of tools to adopt in every DevOps SDLC phase. Among all the security activities, application security testing is one of the commonly used DevSecOps measures. It aims to find the weak points in your source code and overall application security which may be exploited by malicious users. There are three major types of automation tools for this activity: static application security testing (SAST) tools, dynamic application security testing (DAST) tools, and interactive application security testing (IAST) [12].

Activities	**Phase**	**Tool to adopt**
Threat modeling	Plan	Threat modeling tool
Security code development	Develop	IDE
Static code scan before commit	Develop	IDE security plugins
Code commit scan	Develop	Source code repository security plugin
Static application security test and scan	Build	SAST tool
Dependency vulnerability checking	Build	Dependency checking / BOM checking tool
Dynamic application security test and scan	Test	DAST tool or IAST tool
Manual security testing (such as penetration test)	Deploy	Varies tools and scripts (may include network security test tool)
Post-deployment security scan	Deploy	Security compliance tool
Operational dashboard	Operate	Backup
System Security monitoring	Monitor	Information Security Continuous Monitoring (ISCM)

Table 2: Security Activities and Tool categories Summary [11]

SAST, also referred to as "white-box testing", tests the application from the inside out by scanning the source code without executing the program. It will analyze the inner mechanism of the program: everything from language, dependencies, method calls, and execution order, and compare them with a database of known vulnerabilities. One instance of such kind of tool is SonarQube [13]. It is an open-source tool that can automatically perform code reviews and detect bugs, code smells, and vulnerabilities in your code. It fits well with most existing automation tools such as Jenkins, Github, and Azure DevOps. Integrating into the team's workflow, it can provide continuous code review across all project branches and pull requests.

DAST, on the other hand, is a black-box security testing method that tries to test the application from the outside and examine it by attacking it in its running state. One such kind of tool is OWASP Zed Attack Proxy (ZAP). ZAP is an open-source penetration testing tool. It works as a "man in the middle proxy" and 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[14].

Compared to DAST, SAST tools possess many advantages such as requiring less time and fewer costs as almost all the tests can be automated. They can also detect application defects at earlier stages, which leads to lower costs for remediation. However, SAST tools also suffer some drawbacks such as high positive rates in their reports and limited code coverage [15], while DAST tools are less skeptical to these problems as they test the running application from the outside and focus on real and identifiable vulnerabilities [12].

IAST appears in an attempt to combine some of the best characteristics of SAST and DAST. It works like DAST while the application is running but proceeds testings from inside of the application. It works by deploying agents in the application post-build. The agent observes the application's operation and analyzes traffic flow to identify security vulnerabilities [16].

3.3 Measurement and sharing

Measurement and sharing are other two aspects to consider when implementing DevSecOps. Measurement requires companies to implement measurement metrics related to security. Nora Tomas et al. gave some instances of metrics such as time spent correcting mistakes and the number of developers gone through security training [6]. Sharing means sharing security knowledge between the security team and the rest of the organization. Quite similar to the culture aspect, it also emphasizes security education and improving security awareness for all teams.

4 Discussion - Future trend 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* 3.

Lacking adequate security tools is one of the main challenges faced by current companies when adopting DevSecOps. According to Roshan N.Rajapakse et al. [17], many tools have problems such as high false-positive rates, costing too much time and resources, and demanding too manual operations. However, these challenges contain great opportunities and we believe we will witness the

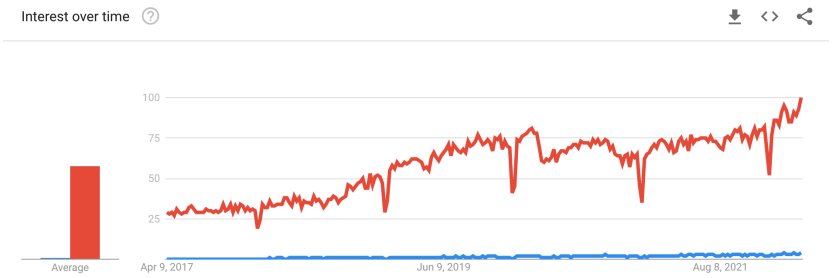

Figure 2: Google trend for term *DevOps* and *DevSecOps* in the last five years

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

Table 3: Number of google scholar search results with term DevSecOps

growth of AI-driven security tools attempting to overcome the deficits of current tools.

One existing example is an AI-powered penetration testing tool Pentoma [18]. Compared to traditional penetration testing tools, it will use ML algorithms to incorporate new vulnerability discoveries at each hacking attempt, thus continuously improving and expanding threat detection capability.

AI also shows potential in helping reduce false-positive rates for SAST tools. Bhawna Yadav et al. proposed a way of using machine learning to improve precision when analyzing code vulnerabilities and reduce analysis time [19].

The adoption of AI will improve the performance of existing security tools, with more accurate security analysis and the ability to reveal more potential vulnerabilities. AI can also help further automate DevSecOps and increase its scalability. With the increase of adoption of AI in DevSecOps, tools will be easier to use with fewer manual operation requirements, which could also improve the acceptance of DevSecOps in companies. Undoubtedly, AI will push the development of DevSecOps in the future and bring it to a new level.

5 Conclusion

In this essay, we discussed DevSecOps, how to implement it, and its future trend. The key point of DevSecOps is "shifting left" - to perform security testing early

in the development lifecycle. This helps reduce the delay and cost of security tasks. Regarding how to implement DevSecOps, we talked about it from the four aspects: CAMS. We focused on the culture and automation parts and introduced application security testing tools and important culture rules such as collaboration and sharing knowledge. Compared to DevOps, DevSecOps is an easily neglected field but is growing rapidly and has a promising future. Especially, with the application of AI in this field, we believe there will be more excellent security tools and it will be known and accepted by more people.

References

[1] S. Gunja, "What is devops? unpacking the rise of an it cultural revolution." `https://www.dynatrace.com/news/blog/what-is-devops/`, 2021.

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

[3] A. Koskinen, "Devsecops: building security into the core of devops," 2019.

[4] V. Blomberg, "Adopting devops principles, practices and tools. case: Identity & access management," *in practice*, vol. 29, no. 6, pp. 1–14, 2019.

[5] L. Gaines, "Cost of fixing vs. preventing bugs." `https://www.coderskitchen.com/cost-of-fixing-vs-preventing-bugs/`, February 2021.

[6] N. Tomas, J. Li, and H. Huang, "An empirical study on culture, automation, measurement, and sharing of devsecops," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, 2019.

[7] M. Sánchez-Gordón and R. Colomo-Palacios, *Security as Culture: A Systematic Literature Review of DevSecOps*, p. 266–269. New York, NY, USA: Association for Computing Machinery, 2020.

[8] P. Perera, R. Silva, and I. Perera, "Improve software quality through practicing devops," in *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 1–6, 2017.

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

[10] H. Myrbakken and R. Colomo-Palacios, "Devsecops: A multivocal literature review," in *Software Process Improvement and Capability Determination* (A. Mas, A. Mesquida, R. V. O'Connor, T. Rout, and A. Dorling, eds.), (Cham), pp. 17–29, Springer International Publishing, 2017.

[11] U.S. Department of Defense Chief Information Officer, *DevSecOps Fundamentals Guidebook: DevSecOps Tools amp; Activities*. Oct 2021.

[12] S. J. Bigelow, "SAST vs. DAST vs. IAST: Security testing tool comparison." https://www.techtarget.com/searchsoftwarequality/tip/SAST-vs-DAST-vs-IAST-Security-testing-tool-comparison, June 2021.

[13] SunarQube, "Code security, for developers." https://www.sonarqube.org/features/security/.

[14] O. ZAP, "Getting started." https://www.zaproxy.org/getting-started/.

[15] J. Li, "Vulnerabilities mapping based on owasp-sans: a survey for static application security testing (sast)," *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pp. 2516–0281, 2020.

[16] J. Peterson, "All about IAST – Interactive Application Security Testing." https://www.mend.io/resources/blog/iast-interactive-application-security-testing/, July 2020.

[17] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," *Information and Software Technology*, vol. 141, p. 106700, 2022.

[18] Pentoma, "Automate your penetration testing tasks." https://se.works/product/pentoma.

[19] B. Yadav, G. Choudhary, S. K. Shandilya, and N. Dragoni, "Ai empowered devsecops security for next generation development," in *Frontiers in Software Engineering* (G. Succi, P. Ciancarini, and A. Kruglov, eds.), (Cham), pp. 32–46, Springer International Publishing, 2021.