



ወልደያ ዩኒቨርሲቲ
Woldia University
Open Mind. Open Eyes!

WOLDIA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

Software Engineering Tools And Practice

Assignment one Development security operations (DevSecOps)

Individual assignment

Name: Abel Amare

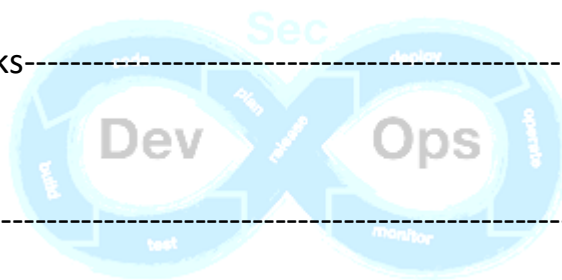
ID: 1305884

Submitted date: March 20, 2024

Submitted to Esmael

Table of contents

What is devsecops-----	3
benefits of DevSecOps-----	4
causes of DevSecOps-----	5
How DevSecOps works-----	6
DevSecOps life-cycle-----	8
Tools of DevSecOps.-----	10
Devsecops career path-----	12
Conclusion-----	15



What is DevSecOps?

DevSecOps, which is short for *development, security and operations*, is an application development practice that automates the integration of security and security practices at every phase of the software development lifecycle, from initial design through integration, testing, delivery and deployment.

DevSecOps represents a natural and necessary evolution in the way development organizations approach security. In the past, security was 'tacked on' to software at the end of the development cycle, almost as an afterthought. A separate security team applied these security measures and then a separate quality assurance (QA) team tested these measures. This ability to handle security issues was manageable when software updates were released just once or twice a year. But as software developers adopted Agile and [DevOps](#) practices, aiming to reduce software development cycles to weeks or even days, the traditional 'tacked-on' approach to security created an unacceptable bottleneck.

DevSecOps integrates application and infrastructure security seamlessly into Agile and DevOps processes and tools. It addresses security issues as they emerge, when they're easier, faster, and less expensive to fix, and before deployment into production.

Additionally, DevSecOps makes application and infrastructure security a shared responsibility of development, security and IT operations teams, rather than the sole responsibility of a security silo. It enables “software, safer, sooner”—the DevSecOps motto—by automating the delivery of secure software without slowing the software development cycle.

Benefits of DevSecOps

The two main benefits of DevSecOps are speed and security. Therefore, development teams deliver better, more-secure code faster and cheaper.

“The purpose and intent of DevSecOps is to build on the mindset that everyone is responsible for security with the goal of safely distributing security decisions at speed and scale to those who hold the highest level of context without sacrificing the safety required,” describes Shannon Lietz, co-author of the “DevSecOps Manifesto.”

Rapid, cost-effective software delivery

When software is developed in a non-DevSecOps environment, security problems can lead to huge time delays. Fixing the code and security issues can be time-consuming and expensive. The rapid, secure delivery of DevSecOps saves time and reduces costs by minimizing the need to repeat a process to address security issues after the fact.

This process becomes more efficient and cost-effective since integrated security cuts out duplicative reviews and unnecessary rebuilds, resulting in more secure code.

Improved, proactive security

DevSecOps introduces cybersecurity processes from the beginning of the development cycle. Throughout the development cycle, the code is reviewed, audited, scanned and tested for security issues. These issues are addressed as soon as they are identified. Security problems are fixed before additional dependencies are introduced. Security issues become less expensive to fix when protective technology is identified and implemented early in the cycle. Additionally, better collaboration between development, security and operations teams improves an organization's response to incidences and problems when they occur. DevSecOps practices reduce the time to patch vulnerabilities and free up security teams to focus on higher value work. These practices also ensure and simplify compliance, saving application development projects from having to be retrofitted for security.

Accelerated security vulnerability patching

A key benefit of DevSecOps is how quickly it manages newly identified security vulnerabilities. As DevSecOps integrates vulnerability scanning and patching into the release cycle, the ability to identify and patch common vulnerabilities and exposures (CVE) is diminished. This capability limits the window that a threat actor has to take advantage of vulnerabilities in public-facing production systems.

Automation compatible with modern development

Cybersecurity testing can be integrated into an automated test suite for operations teams if an organization uses a [continuous integration/continuous delivery](#) pipeline to ship their software. Automation of security checks depends strongly on the project and organizational goals. Automated testing can ensure that incorporated software dependencies are at appropriate patch levels, and confirm that software passes security unit testing. Plus, it can test and secure code with static and dynamic analysis before the final update is promoted to production.

A repeatable and adaptive process

As organizations mature, their security postures mature. DevSecOps lends itself to repeatable and adaptive processes. DevSecOps ensures that security is applied consistently across the environment, as the environment changes and adapts to new requirements. A mature implementation of DevSecOps will have a solid automation, configuration management, orchestration, [containers](#), immutable infrastructure and even [serverless](#) compute environments.

Causes of devsecops



DevSecOps, the integration of security practices into the DevOps process, emerged in response to several software engineering challenges.

1. **Lack of Security Expertise:** Developers often lack deep knowledge of security and compliance. This gap can lead to vulnerabilities in software systems.
2. **Collaboration Barriers:** Traditional silos between security, development, and operations teams hinder effective collaboration. DevSecOps seeks to break down these barriers, encouraging cross-functional team to work together seamlessly.
3. **Tooling Challenges:** Integrating security tools into existing DevOps pipelines can be complex. Finding the right tools and ensuring their proper usage within the workflow is crucial for successful DevSecOps adoption.
4. **Cultural Shift:** Organizations must foster a security-conscious culture. This involves promoting security awareness, encouraging secure coding practices, and emphasizing the importance of security throughout the software development process.
5. **Quality Impact:** Sometimes, security takes a backseat to other priorities during development. DevSecOps aims to address this by making security an integral part of the quality assurance process.

[individual assignment]

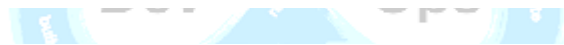
In summary, DevSecOps was initiated to tackle these challenges by embedding security practices into the DevOps methodology, ultimately enhancing software security and reliability.

What are the Phases of DevSecOps?

In the face of a notable surge in security breaches, organizations recognize the importance of prioritizing a security-first approach. As experts anticipate an escalation in hacker tactics, adopting a security-first mindset becomes indispensable for enterprises spanning various industries.

Organizations possess diverse tools and solutions for integrating security into their Software Development Lifecycle (SDLC). Due to the varying structures, processes, toolsets, and overall maturity levels of different SDLCs, there is no universally applicable blueprint for implementing DevSecOps.

DevSecOps phases extend modern DevOps methodologies by integrating security processes and automation into the development pipeline. This allows development teams to maintain the momentum of rapid and continuous delivery while enhancing the security of software assets. The DevSecOps model pipeline adheres to the well-known DevOps “infinity loop” structure, incorporating additional steps to protect code security before, during, and after its deployment to production.



Plan

The planning phases of DevSecOps integration are the least automated, involving collaboration, discussion, review, and a strategy for security analysis. Teams must conduct a security analysis and develop a schedule for security testing that specifies where, when, and how it will carry it out.

IriusRisk, a collaborative threat modeling tool, is a well-liked DevSecOps planning tool. There are also tools for collaboration and conversation, like Slack, and solutions for managing and tracking issues, like Jira Software.

Code

Developers can produce better secure code using DevSecOps technologies during the code phase. Code reviews, static code analysis, and pre-commit hooks are essential code-phase security procedures.

Every commit and merge automatically starts a security test or review when security technologies are directly integrated into developers’ existing Git workflow. These technologies support different integrated development environments and many programming languages. Some popular security tools include PMD, Gerrit, SpotBugs, CheckStyle, Phabricator, and Find Security Bugs.

Build

The ‘ build ‘ step begins once developers develop code for the source repository. The primary objective of DevSecOps build tools is automated security analysis of the build output artifact.

[individual assignment]

Static application software testing (SAST), unit testing, and software component analysis are crucial security procedures. Tools can be implemented into an existing [CI/CD pipeline](#) to automate these tests.

Dependencies on third-party code, which may come from an unidentified or unreliable source, are frequently installed and built upon by developers. In addition, dependencies on external code may unintentionally or maliciously involve vulnerabilities and exploits. Therefore, reviewing and checking these dependencies for potential security flaws during the development phase is crucial.

The most popular tools to create build phase analysis include Checkmarx, SourceClear, Retire.js, SonarQube, OWASP Dependency-Check, and Snyk.

Test

The test phase is initiated once a build artifact has been successfully built and delivered to staging or testing environments. Execution of a complete test suite requires a significant amount of time. Therefore, this stage should fail quickly to save the more expensive test tasks for the final stage.

Dynamic application security testing (DAST) tools are used throughout the testing process to detect application flows such as authorization, user authentication, endpoints connected to APIs, and SQL injection.

Multiple open-source and paid testing tools are available in the current market. Support functionality and language ecosystems include BDD Automated Security Tests, Boofuzz, JBro Fuzz, OWASP ZAP, SecApp suite, GAUNTLET, IBM AppScan, and Arachi.

Release

The application code should have undergone extensive testing when the DevSecOps cycle is released. The stage focuses on protecting the runtime environment architecture by reviewing environment configuration values, including user access control, network firewall access, and personal data management.

One of the main concerns of the release stage is the principle of least privilege (PoLP). PoLP signifies that each program, process, and user needs the minimum access to carry out its task. This combines checking access tokens and API keys to limit owner access. Without this audit, a hacker can come across a key that grants access to parts of the system that are not intended. In the release phase, configuration management solutions are a crucial security component. Reviewing and auditing the system configuration is then possible in this stage. As a result, commits to a configuration management repository may use to change the configuration, which becomes immutable. Some well-liked configuration management tools include HashiCorp Terraform, [Docker](#), [Ansible](#), [Chef](#), and [Puppet](#).

Deploy

If the earlier process goes well, it's the proper time to deploy the build artifact to the production phase. The security problems affecting the live production system should be addressed during deployment. For instance, it is essential to carefully examine any configuration variations between the current production environment and the initial staging and development settings. In addition, production TLS and DRM certificates should be checked over and validated in preparation for upcoming renewal.

The deploy stage is a good time for runtime verification tools such as Osquery, Falco, and Tripwire. It can gather data from an active system to assess if it functions as intended.

[individual assignment]

Organizations can also apply chaos engineering principles by testing a system to increase their confidence in its resilience to turbulence. Replicating real-world occurrences such as hard disc crashes, network connection loss, and server crashes is possible.

Operation

Another critical phase is operation, and operations personnel frequently do periodic maintenance. Zero-day vulnerabilities are terrible. Operation teams should monitor them frequently. DevSecOps integration can use IaC tools to protect the organization's infrastructure while swiftly and effectively preventing human error from slipping in.

Monitor

A breach can be avoided if security is constantly being monitored for abnormalities. As a result, it's crucial to put in place a robust continuous monitoring tool that operates in real-time to maintain track of system performance and spot any exploits at an early stage.

DevSecOps life cycle



DevSecOps is an approach towards software development that focuses on incorporating security into the dev-ops process. It has three main components, namely Development (dev), Security (sec), and Operations (ops).

In dev-ops, the dev phase covers all activities related to coding and testing, while the sec is responsible for maintaining secure databases, policies, and procedures.

1. Code Analysis

DevSecOps incorporates code analysis as an integral part of its DevSecOps flow. Code analysis is used to identify any anomalous traits in the production environment that could compromise the system. It is done by analyzing the codebase in order to detect bugs or vulnerabilities that could lead to possible security threats before releasing the code into full production.

The result of code analysis helps development teams efficiently take corrective measures and strengthen their DevSecOps operations. In addition, it also helps to identify best coding practices while improving upon existing ones with regard to security standards.

2. Change Management

Change management is a key component of DevSecOps and is essential for successful deployments. It involves the systematic analysis of changes being made, especially when larger projects are taking place, in order to ensure that these changes have been correctly implemented at every step in the DevSecOps flow.

The basic idea behind DevSecOps changes management is that changes should be tested against performance standards or acceptance criteria before they can be deployed or published. This allows project teams to quickly identify and mitigate potential issues without disrupting the DevSecOps pipeline, increasing market speed and reducing risk levels.

3. Compliance Management

Compliance management is an essential component of DevSecOps flow. It helps ensure that organizations adhere to various regulatory frameworks & industry-specific requirements related to software development. Such as, the ability to detect and respond to security threats.

As part of DevSecOps, compliance management allows organizations to continuously monitor changes to their codebase for vulnerabilities. It also enables organizations to automate testing to detect any issues sooner rather than later.

This helps organizations remain compliant with the expectations set by the environments they operate in, allowing them to invest more resources into innovation while running secure solutions efficiently.

4. Threat Modeling

One important component of DevSecOps is threat modeling, which looks at how data flows throughout DevSecOps and how security threats can be identified and mitigated. Threat modeling occurs early in the DevSecOps flow, serving as a preventative measure to reduce or eliminate potential vulnerabilities or risks within the DevSecOps ecosystem.

It involves analyzing application design, architecture, and implementation details to assess cyber threats from external or internal sources, ensuring that best practices with regard to security are applied during development.

While human error or malicious behavior cannot always be prevented, DevSecOps threat modeling minimizes the risk of exploitation through security assurance checks and addressing weak points in the system before launch.

5. Security Training

Security training helps build stronger cyber defenses and guards against malicious actors' attempts to attack your organization. It should be part of DevSecOps from the start, beginning with educating team members on the DevSecOps process, tools, best practices, and security policies.

Additionally, security training can be used to reinforce DevSecOps principles as they evolve over time and create an ongoing culture of data privacy and cyber safety. Knowledgeable team members are better equipped to recognize potential threats in the DevSecOps flow before they become a problem.

Security training as part of DevSecOps will ensure your organization is up-to-date with the latest best practices and industry standards for protecting your data.

Devsecops tools

1. Software Composition Analysis (SCA)

Given the fact that open source software makes up over 90% of the codebase of modern applications, SCA has become an indispensable DevSecOps tool.

Software composition analysis (SCA) tools scan applications to detect and address issues (security vulnerabilities, problematic OSS licenses, and quality issues) in open source code. SCA solutions also offer reporting functionality, including the ability to generate a software bill of materials. If and when SCA does identify a vulnerability, it provides a host of information (including severity score, inclusion path, and remediation guidance) to help users properly address the issue.

For the open source license compliance use case, SCA inventories the different licenses involved in your code, flagging any components with licenses that violate an organization's compliance policies.

Finally, modern SCA tools also help teams implement the key DevSecOps principle of delivering *quality* software. SCA offers code quality and provenance checks, helping users identify and upgrade outdated and/or poorly maintained software components.

2. Static Application Security Testing (SAST)

SAST refers to a set of tools that scan codes (source code, binary code, byte code) in a non-running (read: static) state. SAST flags weaknesses in the code it scans, effectively surfacing common issues like CWE-79 (cross-site scripting), buffer overflow errors, SQL Injection, and more.

Much like SCA, SAST flags vulnerabilities and offers remediation guidance. Both tools analyze source code/binaries as opposed to running applications. And, both SCA and SAST are frequently used during the “build” stage of the software development lifecycle, in line with the “shift-left” principle of conducting security testing as early as possible in the SDLC.

There are several significant differences between SCA and SAST, however. While SCA identifies vulnerabilities in open source code, SAST detects vulnerabilities in proprietary code. And, as you might expect, open source license compliance is *not* a SAST use case. DevSecOps teams often use SCA and SAST in a complementary manner.

3. Dynamic Application Security Testing (DAST)

In contrast to SAST and SCA, DAST (Dynamic Application Security Testing) tests for vulnerabilities in a running application. As such, it's used later in the software development lifecycle.

DAST does not require access to source code. Instead, DAST tools detect vulnerabilities in a running application by (safely) injecting malicious inputs to identify potential security vulnerabilities within the application. A DAST tool will make HTTP requests and uncover issues like SQL injections, OS injections, and cross-site scripting errors. It also finds bugs that are

important to application security contexts, like security headers, cookie safety, content security policies, and X-Frame-Options.

There's no language dependency with DAST tools because they test the running app, however you compile it. DAST also takes into account the context of how the application works: It tests the running application with bad inputs to see how the application behaves. Security teams often use DAST tools as part of their application security suites along with SAST, SCA, and more.

4. Automated Testing Tools

The days of large, dedicated QA teams are a thing of the past for organizations with successful DevSecOps implementations. As the U.S. government's DevSecOps Fundamentals Guidebook puts it: "Testing is about automation, and testers will need to become coders of that automation."

Although some manual testing work will still be required — it's not possible to automate every part of every test — the majority can be automated. For example:

- **Unit tests:** Unit tests analyze individual units of code to make sure they perform as expected. Unit testing tools tend to be language-specific.
- **Integration tests:** Integration tests are performed after unit tests and deal with the interaction between units of code. Again, many of these tests are language-specific.
- **System tests:** System tests are performed after integration tests and analyze the entire application. System testing tools analyze areas like usability, reliability, scalability, and more.

Performance testing, regression testing, and acceptance testing are also among the areas that can be automated.

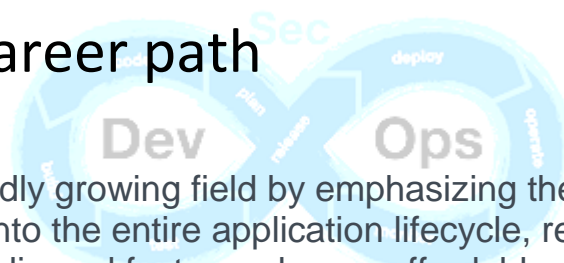
5. Issue Tracking System

The final tool we'll discuss is one that most teams are likely already familiar with: issue tracking software. Issue tracking systems support several key DevSecOps phases and activities.

Key characteristics of issue tracking tools include:

- **Automation:** Improves engineering efficiency by automating processes like closing issues, notifying customers, assigning issues, and more
- **Issue resolution tracking and history:** Provides visibility and structure to enable efficient bug management. Also creates a record of activities related to issue resolution.
- **Change management:** Equips stakeholders with visibility into new feature development. Offers interactive workflows and roadmaps to support planning and development.
- **Prioritization management:** Enables teams to easily (i.e. drag and drop) prioritize different fixes and activities so that they continuously address the most important items
- **Automated reporting capabilities:** Offers a consolidated view of created vs. resolved issues, resolution velocity, development velocity, and other important metrics

Devsecops career path



DevSecOps is a rapidly growing field by emphasizing the philosophy of integrating security into the entire application lifecycle, resulting in better, more secure code delivered faster and more affordably. As a result, companies are increasingly looking for skilled DevSecOps engineers to join their teams. In this article, we'll provide a comprehensive DevSecOps career path to becoming a Skilled DevSecOps engineer for aspiring and current IT professionals interested in this field.

What is a DevSecOps Engineer?

A DevSecOps engineer is a skilled security professional who ensures that security is seamlessly and effectively integrated with the Software Development Life Cycle (SDLC). This entails identifying potential security vulnerabilities and devising robust mitigation strategies to prevent such risks from materializing.

Additionally, DevSecOps engineers are responsible for monitoring security threats, implementing security controls, and ensuring compliance with relevant security standards and regulations. Overall, these professionals are vital in ensuring that security is at the forefront of the SDLC and that security best

[individual assignment]

practices are consistently followed throughout the development and deployment process.

The DevSecOps Career Path

The DevSecOps career path starts with a solid foundation in software development. Many DevSecOps engineers start as software developers or system administrators before transitioning to a DevSecOps role. Relevant certifications, such as [Certified DevSecOps Professional \(CDP\)](#) and [Certified DevSecOps Expert \(CDE\)](#), can also help you to advance in this career path. You can also reach for leadership roles in DevSecOps through certifications like [Certified DevSecOps Leader \(CDL\)](#)

DevSecOps Engineer Skills

To become a skilled DevSecOps engineer, you must have various technical and soft skills. Some of the best DevSecOps engineer skills are:

1. Strong understanding of security concepts, including threat modeling, risk assessment, and vulnerability management.
2. Knowledge of the SDLC and experience integrating security best practices at every process stage.
3. Familiarity with automation tools and scripting languages like Python and PowerShell.
4. Understanding cloud security principles, including secure architecture design and configuration management.
5. Knowledge of container security principles, such as Docker and Kubernetes.
6. Experience with DevOps practices, such as continuous integration and delivery (CI/CD) and infrastructure as code (IaC).
7. Familiarity with compliance frameworks and regulations, such as PCI-DSS, HIPAA, and GDPR.
8. Strong problem-solving skills, including analyzing complex security issues and developing effective solutions.
9. Effective communication skills, including working collaboratively with cross-functional teams.
10. A passion for continuous learning and keeping up with the latest security trends and technologies.

DevSecOps Engineer Roles and Responsibilities

DevSecOps engineer roles and responsibilities are various tasks, including:

- Integrating security into the software development process.

- Identifying potential security risks and developing strategies to mitigate them.
- Implementing security controls.
- Monitoring security threats.
- Ensuring compliance with security standards and regulations.
- Collaborating with developers, system administrators, and other stakeholders to ensure that security is integrated into the development process

DevSecOps Engineer Requirements

DevSecOps engineer requirements are several, and some of them are as follows:

- Early detection of security vulnerabilities
- Faster deployment of secure software
- Greater collaboration between development, security, and operations teams
- Improved compliance with security standards and regulations
- Greater visibility into security risks and threats

How to Become a DevSecOps Engineer?

To become a DevSecOps engineer, you need to have a solid foundation in software development and security principles. You can start by obtaining a degree in computer science, information technology, or a related field.

Relevant certifications such as [Certified DevSecOps Professional CDP](#) can also help demonstrate your security expertise.

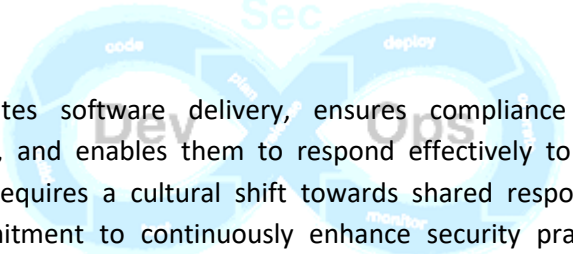
What Does a DevSecOps Engineer Do?

DevSecOps engineers are required to be capable of efficiently implementing a range of DevSecOps best practices, including:

- Implement security early and often in the SDLC to identify and mitigate risks at every stage of the process.
- Establish a security culture across the organization and ensure all stakeholders know their responsibilities.
- Automate as much of the security testing and deployment process as possible to reduce the risk of human error.
- Adopt a risk-based approach to security, focusing on the most critical assets and vulnerabilities.

- Use infrastructure as code (IaC) to enable more efficient and consistent deployment of secure environments.
- Conduct regular security assessments and penetration testing to identify vulnerabilities and improve security posture.
- Encourage collaboration between security, development, and operations teams to share knowledge and best practices.
- Monitor the environment for security threats and respond quickly to incidents or breaches.
- Use a security-focused DevOps toolchain to enable seamless integration of security testing and deployment processes.
- Ensure compliance with security standards and regulations, such as PCI-DSS, HIPAA, and GDPR, by building security into the SDLC.

Conclusion:



security posture, accelerates software delivery, ensures compliance with regulations, fosters collaboration across teams, and enables them to respond effectively to emerging security threats. Implementing DevSecOps requires a cultural shift towards shared responsibility for security within organizations and a commitment to continuously enhance security practices in an ever-changing technology landscape. provides insights into the roles and responsibilities within local and international DevSecOps career opportunities as well as the typical career path progression for individuals pursuing a career in DevSecOps. Each section highlights the key aspects of different roles and levels within the field of DevSecOps to help individuals understand the various career opportunities available and plan their career advancement accordingly.

By embedding security throughout the software development lifecycle, DevSecOps aims to create secure, resilient, and compliant software systems. Recognizing the underlying software engineering problems underscores the importance of DevSecOps in enhancing security and mitigating risks in software applications. It has its own principles, tools, and paths of their work.