**Appendix A: DevSecOps concepts you need to know**

 DevSecOps means thinking about application and infrastructure security from the start. It also means automating some security gates to keep the DevOps workflow from slowing down. Selecting the right tools to continuously integrate security, like agreeing on an integrated development environment (IDE) with security features, can help meet these goals. However, effective DevOps security requires more than new tools—it builds on the cultural changes of DevOps to integrate the work of security teams sooner rather than later. *DevOps security is built for containers and microservices.* Rather, security must be continuous and integrated at every stage of the app and infrastructure life cycle. DevSecOps means building security into app development from end to end.





**DevOps security is automated**

To do: Maintain short and frequent development cycles, integrate security measures with minimal disruption to operations, keep up with innovative technologies like containers and microservices, and all the while foster closer collaboration between commonly isolated teams—this is a tall order for any organization. All of these initiatives begin at the human level—with the ins and outs of collaboration at your organization—but the facilitator of those human changes in a DevSecOps framework is automation.

**DevOps security is built for containers and microservices**

The greater scale and more dynamic development and deployment enabled by containers have changed the way many organizations innovate. Because of this, DevOps security practices must adapt to the new landscape and align with container-specific security guidelines.

Cloud-native technologies don’t lend themselves to static security policies and checklists. Rather, security must be continuous and integrated at every stage of the app and infrastructure life cycle.

**Container Technology Stack**

<https://csrc.nist.gov/pubs/ir/8176/final>



DevSecOps means building security into app development from end to end. This integration into the pipeline requires a new organizational mindset as much as it does new tools. With that in mind, DevOps teams should automate security to protect the overall environment and data, as well as the continuous integration/continuous delivery process—a goal that will likely include the security of microservices in containers.

**Behavior-Driven Development (BDD)** is an approach to software development that focuses on how software should behave from the end user’s perspective. It works on the principle that we can design better software that meets users’ needs by understanding how they interact with a system.

**Continuous Delivery (CD)** is the practice of automatically building, testing, and deploying code changes. It allows for quicker feedback and delivery of new features and bug fixes to users. Continuous delivery typically works with continuous integration, which automatically merges code changes into a shared codebase.

entails automating the build and testing of code every time a team member commits changes to version control. This approach lets developers quickly detect errors, locate and fix them easily, and avoid spending hours or days debugging code.

**DevOps Model**: The DevOps model is a relatively new approach focusing on collaboration and communication between developers and operations teams. It helps speed up the software development process by eliminating bottlenecks and allowing for continuous delivery of code changes.

is a set of practices that combine software development (Dev) and IT operations (Ops). It aims to shorten the systems development lifecycle and provide continuous delivery with high software quality. In DevOps, security gets relegated to a mere afterthought.

**DevSecOps pipeline**: A DevSecOps pipeline is a set of automated processes that allow for the continuous delivery (CD) and continuous integration (CI) of code changes while ensuring security controls are in place. A DevSecOps pipeline aims to provide faster feedback loops, shorter development cycles, and improved communication between Development, Security, and Operations teams.

**DevSecOps** is a security-focused extension of the DevOps model. It adds security into the mix, so that development, operations, and security teams can work together from the start to build secure applications and systems. DevSecOps aims to weave security into the fabric of the software development process. This way, it becomes second nature for developers and operations teams to think about safety at every stage of the application development lifecycle.

**Distributed Version Control System (DVCS)** allows each user to have a complete local copy of the entire repository. It enables users to work offline and later merge their changes into the main repository. Often, large teams leverage DVCSs as it is practical only for a few developers to work on the same central server simultaneously.

**Dynamic Application Security Testing (DAST)** is a type of security testing performed on web applications. It is a process of testing web applications for vulnerabilities by executing attacks against them. DAST helps find vulnerabilities such as SQL injection, cross-site scripting, and session hijacking.

**Fuzz testing** is software testing that looks for errors in software by providing invalid, unexpected, or random data as inputs to the software. Fuzz testing aims to find bugs that could cause crashes or security vulnerabilities.

[**False Positive**](https://www.guardrails.io/blog/false-positives-and-false-negatives-in-information-security/)is an incorrect or inaccurate security alert generated by a tool or system, indicating the presence of a security vulnerability or threat that does not actually exist. This can waste valuable time and resources of security professionals and divert their attention from real security issues.

**Interactive Application Security Testing (IAST)** is a type of security testing that combines static and dynamic analysis techniques to identify vulnerabilities in web applications. IAST tools can instrument the application to provide visibility into its execution and identify potential security issues.

**Identity and Access Management (IAM)** is a process for managing digital identities. IAM systems store, manage and distribute digital identities. They are used to authenticate and authorize users to access digital resources. The IAM process includes creating and managing user accounts, assigning roles and permissions, and auditing access to improve security, reduce costs, and increase efficiency. IAM is a critical component of security for organizations of all sizes.

**Mobile Applications Security Testing (MAST)** is the process of assessing the security of a mobile app. It includes testing for vulnerabilities and threats that could jeopardize the safety of app users or the data stored on their devices. It helps identify potential risks and weaknesses in an app before its public release.

**Security posture** is the term used to describe the security of an organization or individual. It includes both the physical and cyber security of an organization. The goal of having a good security posture is to be proactive in identifying and mitigating risks before they happen. Teams can implement policies and procedures and train employees to identify and respond to potential threats.

entails keeping security information and tools isolated. The rationale behind this approach is that separately maintaining security information and tools makes it more difficult for attackers to gain a holistic view of an organization’s security posture. It will blunt their ability to exploit weaknesses.

**Software Development LifeCycle (SDLC)** is the process of creating and maintaining software. It includes all phases of software development, from planning and requirement gathering to design, implementation, testing, and deployment.

**Shift-left** describes moving to test earlier in the software development cycle. It contrasts with the true left, which refers to moving testing to the beginning of the development cycle. Shift-left emphasizes the need for collaboration between developers and testers throughout the software development process.

**Software Composition Analysis (SCA)** analyzes software to identify its parts and their functions. Teams can use this information to assess the software’s security, quality, and compliance. SCA can be manually done or automated.

**Static Application Security Testing (SAST)** looks for security vulnerabilities in the source code of compiled binaries. It can be done manually or using automated tools. The goal of SAST is to find security vulnerabilities in software before its deployment. It is in contrast to Dynamic Application Security Testing (DAST), which looks for security vulnerabilities while the software runs.

**Security Champions** are developers who have shown a security interest and help bridge the gap between DevOps and your application security expertise. They assist in bridging long-standing schisms among teams and breaking down obstacles.

**Test-driven Development (TDD)** is a methodology that advocates writing tests before production code. The idea behind TDD is that by writing tests first, we can derive the requirements for our production code from them. It results in a cleaner and more maintainable code. TDD has become a popular development methodology in recent years, especially among developers who work on large and complex codebases.

**Vulnerability Assessments (VA)** are used to identify, quantify, and prioritize vulnerabilities in their systems and networks. Vulnerability assessments provide organizations with a roadmap for improving their security posture by first identifying the most critical vulnerabilities.

**Version Control System (VCS)** is software that allows you to track the changes made to files over time. It is helpful for developers who need to work concurrently on the same code base but want to avoid merge conflicts.

### **Key DevSecOps Terms**

### 1. Attack Surface

### The attack surface refers to the potential vulnerabilities within a system that an attacker can exploit. It represents the exposure that the network has to potential threats. Every network interaction point is part of the attack surface.

### 2. Automation

### In DevSecOps, automation refers to applying technology—scripts, bots, algorithms, etc.—to execute security tasks throughout the software development life cycle. Automation increases efficiency, accuracy, and consistency while reducing human error.

### 3. Chain of Custody

### The chain of custody pertains to the record of who possessed digital evidence at a specific time. The chain of custody must be maintained to ensure the evidence has not been altered and its authenticity can be verified.

### 4. CI/CD (Continuous Integration and Continuous Delivery)

### CI/CD is a software development practice where developers frequently integrate code changes into a shared repository. Software changes are automatically built, tested, and deployed to production. This methodology necessitates higher levels of security to reduce the possibility of disruption.

### 5. Code Dependencies

### Code dependencies are the external libraries, frameworks, and modules your code requires. If not managed correctly, these dependencies can introduce vulnerabilities into your codebase.

### 6. Compliance

### In DevOps and security, compliance refers to an organization's adherence to external regulations, standards, best practices, and internal company policies.

### 7. Configuration Drift

### Configuration drift occurs when the configuration of a system changes without being tracked or approved. This drift can lead to security vulnerabilities as the organization’s scope broadens.

### 8. Containerization

### Containerization is a method of packaging software so it can be run in isolated environments. Containers are self-contained, including all dependencies necessary to run the software, making them portable and secure.

### 9. Data Breach

### A data breach occurs whenever there is unauthorized access to or disclosure of sensitive information. This can happen when a malicious attacker gains system access, or an authorized user mishandles data.

### 10. Data Loss Prevention (DLP)

### Data loss prevention is the practice of preventing unauthorized disclosure of sensitive information. This is accomplished through automated tools or restricted access, such as encryption of data in transit and at rest and monitoring and controlling access to data.

### 11. Endpoint Security

### Endpoint security is securing the devices that connect to a network, including laptops, smartphones, tablets, and IoT devices. This is typically achieved through antivirus software, firewalls, and intrusion detection and prevention systems.

### 12. Identity and Access Management (IAM)

### IAM is the practice of managing digital identities and their access to sensitive information and systems. This includes user account provisioning and de-provisioning and the management of access controls.

### 13. Maturity Model

### A maturity model is a framework used to assess an organization's progress in adopting a particular practice or capability. In the context of DevSecOps, it measures the organization's progress in adopting DevSecOps practices and achieving DevSecOps objectives.

### 14. Passwordless Authentication

### This user authentication method does not rely on passwords but uses biometrics, hardware tokens, or one-time passcodes (OTPs) instead. It is considered more secure, as it is independent of users to uphold security standards.

### 15. Penetration Testing

### Also known as pen testing, this is the practice of simulating an attack on a system to identify vulnerabilities. Pen tests can be conducted manually or with automated tools to target individual systems or an entire network.

### 16. Perimeter Security

### Perimeter security is the practice of protecting the boundaries of a network. Typically, it involves firewalls and intrusion detection and prevention systems.

### 17. Risk Management

### In security, risk management involves identifying, assessing, and mitigating risks. This includes the identification of threats and vulnerabilities, as well as the assessment of their impact on the organization.

### 18. Security Information and Event Management (SIEM)

### SIEM is a security management approach that combines security information management (SIM) and security event management (SEM) functions. It gives organizations a real-time view of their security posture and the ability to detect, investigate, and respond to security incidents.

### 19. Security as Code

### This practice involves treating security configurations and policies as code, which can then be managed like any other software asset. It helps ensure consistency across environments and that changes can be tracked over time.

### 20. Security Posture

### This term refers to an organization’s overall state of security, including the effectiveness of its controls and the adequacy of its policies and procedures.

### 21. Shift Left

### Shift Left is a DevOps principle that advocates for including security earlier in the software development process. Traditional software development often treats security as an afterthought or a final step, but DevSecOps redefines this by embedding security from the onset. This proactive approach helps organizations identify and fix security issues in real-time rather than retrospectively. This saves time and resources and can prevent potentially severe damage. "Shifting security left" can thus create a robust product, improve customer trust, and enhance the company's overall reputation.

### Environment and data security

* **Standardize and automate the environment:**Each service should have the least privilege possible to minimize unauthorized connections and access.
* **Centralize user identity and access control capabilities:**Tight access control and centralized authentication mechanisms are essential for securing microservices, since authentication is initiated at multiple points.
* **Isolate containers running microservices from each other and the network:** This includes both in transit and at rest data, since both can represent high-value targets for attackers.
* **Encrypt data between apps and services:**A container orchestration platform with integrated security features helps minimize the chance of unauthorized access.
* **Introduce secure API gateways:** Secure APIs increase authorization and routing visibility. By reducing exposed APIs, organizations can reduce surfaces of attacks.

### CI/CD process security

* **Integrate security scanners for containers:**This should be part of the process for adding containers to the registry.
* **Automate security testing in the CI process:**This includes running security static analysis tools as part of builds, as well as scanning any pre-built container images for known security vulnerabilities as they are pulled into the build pipeline.
* **Add automated tests for security capabilities into the acceptance test process:** Automate input validation tests, as well as verification authentication and authorization features.
* **Automate security updates, such as patches for known vulnerabilities:**Do this via the DevOps pipeline. It should eliminate the need for admins to log into production systems, while creating a documented and traceable change log.
* **Automate system and service configuration management capabilities:** This allows for compliance with security policies and the elimination of manual errors. Audit and remediation should be automated as well.