Carnegie Mellon University
Software Engineering Institute

DEVSECOPS SYSTEM ASSURANCE

Geoffrey Sanders, Robert Ellison, Carol Woody
September 2021

Introduction

DevSecOps pipelines support organizational agility by automating rapid and frequent delivery of secure infrastructure and software to production (Figure 1). Pipelines are complex systems that require tradeoff decisions for each implementation, which commonly introduce risk to the pipeline and the product it delivers. System assurance should be used to manage that risk and maintain confidence in the pipeline and its product. This paper focuses on system assurance for DevSecOps software systems.

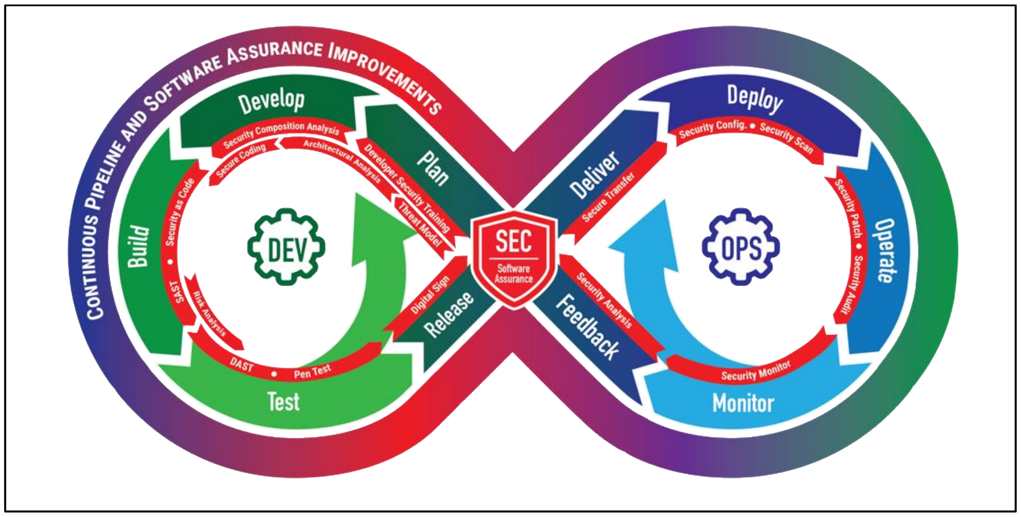

Figure 1: DevSecOps Pipeline (Infinity Diagram)

System assurance is an approach for justifying confidence that a system functions as intended and is free of exploitable vulnerabilities intentionally or unintentionally designed or inserted as part of the system during the lifecycle. While achieving zero vulnerabilities is normally impossible in practice, assurance cases help risk management by reducing their probability and impact to acceptable levels [NDIA 2008; NIST 2015].

Assurance cases convincingly justify to stakeholders that the implemented system meets critical system assurance requirements. The assurance cases comprise a set of claims of critical system assurance properties, arguments that justify the claims (including assumptions and context), and evidence supporting the arguments (Figure 2) [NDIA 2008; Ellison 2008].

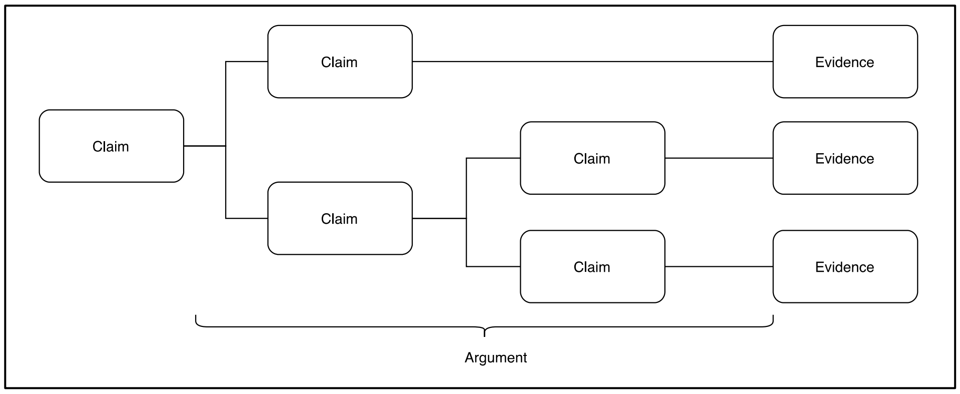

Figure 2: Assurance Case Framework [Ellison 2008]

Assurance case development results in system assurance requirements that flow to the system architecture and product baseline. Systems engineering technical activities applied to those requirements generate assurance cases while also providing technical maturity evidence [NDIA 2008].

DevSecOps System Assurance

The Software Engineering Institute's (SEI) DevSecOps platform independent model (PIM) provides a formal approach and methodology for building a pipeline that can be tailored to an organization's specific requirements while outlining the activities necessary to evolve it. Our work incorporates system assurance to support assurance case development with pipeline data.

The SEI PIM accomplishes system assurance through several functions: software assurance, quality assurance, security assurance, risk management, and audit (Figure 3).

- Software assurance: the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted during its lifecycle, and that the software functions in the intended manner [NIST 2021].
- Quality assurance: a strategic and systematic approach to monitoring the engineering tools, practices, and processes used to ensure the quality of a product under development to assure relevant stakeholders that the product under development will fulfill relevant stakeholder expectations and regulatory requirements. Expectations are ideally explicitly stated through service level agreements, requirements, goals, etc. and not simply implied.
- Security assurance: the measure of confidence that the security features, practices, procedures, and architecture of an information system accurately mediates and enforces security policy [NIST 2021].
- Risk management: the program and supporting processes to manage information security risk to organizational operations (including mission, functions, image, reputation), organizational assets, individuals, other organizations, and the nation. It includes (1) establishing the context for risk-related activities; (2) assessing risk; (3) responding to risk once determined; and (4) monitoring risk over time [NIST 2021].

- Audit: the independent review and examination of records and activities to assess the adequacy of system controls and ensure compliance with established policies and operational procedures [NIST 2021].

Safety and reliability assurance are included in Figure 3 as additional system assurance areas to consider in the future.

Quality assurance and security assurance share similarities but have different goals. Security assurance focuses on failures or properties of the system, such as data integrity, confidentiality, and security development practices. Quality assurance focuses on how development standards, practices, and methods are applied to the system.

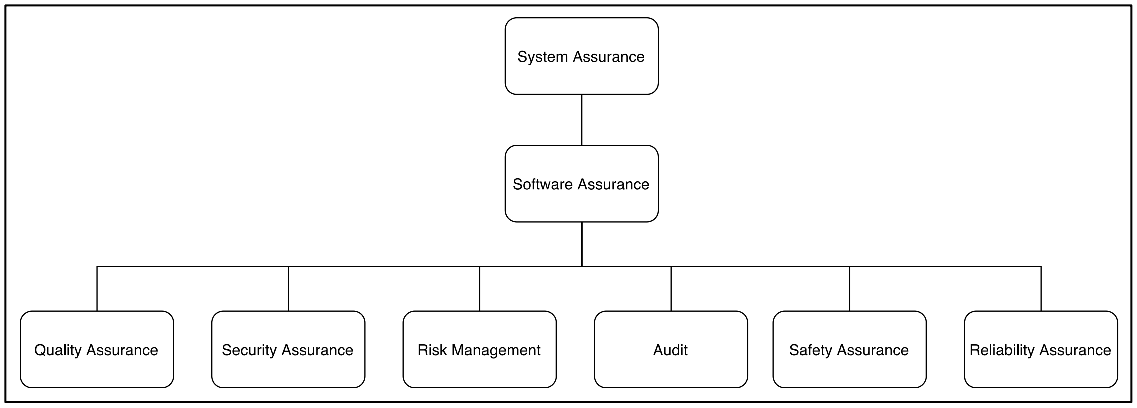

Figure 3: SEI PIM System Assurance

To fulfill these functions, a DevSecOps pipeline must provide a minimum set of system assurance features: assurance case development, assurance case audit, system assurance risk measurement, and assurance case mapping.

- Assurance case development: the system shall be able to capture assurance cases that supply a documented body of *evidence* that provides a convincing and valid *argument* that a specified set of critical *claims* about a system's properties are adequately justified for a given application in a given DevSecOps environment.
- Assurance case audit: the system shall be able to support independent quality assurance reviews, or audit of activities and work products, associated with assurance cases.
- System assurance risk measurement: the system shall be able to support system assurance risk measurement.
- Assurance case mapping: the system shall be able to trace implementation elements to assurance case claims.
- System assurance metrics: the system shall be able to create and track system assurance metrics.

Assurance cases are commonly constructed with Goal Structuring Notation (GSN). Using this notation, claims are classified into subclaims that are supported by evidence while articulating the argumentation strategies adopted for the claim, rationale for the approach, and context in which they are stated [Ellison 2014].

Example System Assurance Thread

Figure 4 illustrates an example thread to demonstrate system assurance with a DevSecOps pipeline. In this thread, we discuss applying incremental threat modeling to each pipeline phase. Phases are represented by rectangles and control gates are represented by red diamonds [DoD 2021a].

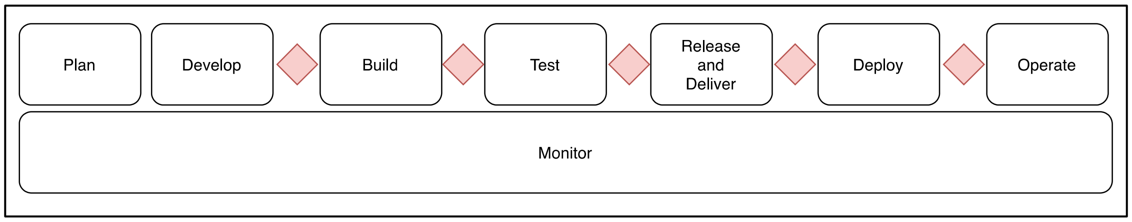

Figure 4: DevSecOps Pipeline Phases [DoD 2021a]

Incremental threat modeling is an agile threat modeling process that assesses every story for new items, such as components, processes, dataflows, or trust boundaries. If new items are introduced, the threat model is extended to identify new threats and their countermeasures. This approach aligns well with agile development where an application may already be in design or development before threat modeling begins [Michlin 2017; Goodwin 2020].

Control gates are manual or automated mandatory actions that determine artifact fitness for promotion to the next pipeline phase. Actions primarily include cyber and operational test and evaluation (OT&E) assessments. It is assumed that control gates established by new DevSecOps teams will require human intervention. However, as the team matures through process improvement, repeatable actions should be automated at control gates wherever possible. Automation is necessary to achieve high performing activities such as continuous Authority to Operate (cATO).

Plan

Ideally, DevSecOps teams begin incremental threat modeling in the Plan phase by applying threat models to a comprehensive understanding of the system, architecture, and system-of-systems. Threat modeling identifies threats, vulnerabilities, and countermeasures that require mitigation during the development sprint. Incremental threat models assume a baseline threat model exists. This model is used to compare against new sprint components.

Assurance case development should also begin in the Plan phase. Assurance claims are established, evidence is identified, and case measurement is defined. Case measurement should continue through all pipeline phases to understand how the sprint affects system assurance at each phase.

Develop

The Develop phase converts requirements into source code, infrastructure, processes, and other artifacts. While these artifacts may directly support threat models and assurance cases, indirect items such as unit tests are also necessary. Multiple team members will need to interpret threat models and assurance cases for the sprint and implement mechanisms that log and collect measurement data [DoD 2021b].

DevSecOps pipeline maintainers may also need to work with developers to establish pipeline changes required for the sprint. Ideally, the pipeline provides application programming interfaces (APIs) and other services that support logging and measurement for each lifecycle iteration

This phase also determines metrics required for the control gate between the Develop and Build phases. Threat models may use metrics such as CVSS (Common Vulnerability Scoring System) scores to measure risk, while assurance cases use confidence levels. Acceptable risk determines artifact control gate transition from Develop to Build.

Build

Building and packaging applications, services, and microservices into artifacts occurs during the build phase. Compiling, linting, documenting, dependency checking, and containerization are common actions that build and package artifacts. While some of these actions may be executed during the Develop phase by developers with an integrated development environment (IDE), most are automated. SAST (static application security test), build configuration control, and auditing are also performed in this phase [DoD 2021b].

Data and metrics from Build phase activities feed threat model and assurance case risk measurement. Example data include CVSS scores, assurance case confidence levels, software patches, and SAST scans. Acceptable risk determines artifact control gate transition from Build to Test.

Test

Continuous, automated testing occurs during the Test phase. Multiple stages can be used during this phase, such as development, system, and pre-production. Common activities for this phase are license compliance checks, dynamic application security testing (DAST), interactive application security testing (IAST), database testing, compliance scans, and system, performance, and regression testing. Manual security testing, such as penetration testing, also occurs in this phase. Manual tests simulate cyberattacks that help identify vulnerabilities, such as logic flaws, that easily escape automated tests [DoD 2021b].

Data and metrics from Test phase activities feed threat model and assurance case risk measurement. Manual testing in this phase will likely generate information in file formats not directly consumable by automation. Processes and procedures should account for manual formats and how they affect control gates. Acceptable risk determines artifact control gate transition from Test to Release and Deliver.

Release and Deliver

Software artifacts are digitally signed and delivered to artifact repositories during the Release and Deliver phase. Repositories may be centralized or distributed depending on mission needs. Multiple repositories may also be implemented that map to Test phase stages. Common activities for this phase are release packaging, artifact replication, operations acceptance, and configuration audit [DoD 2021b].

Data and metrics from Release and Deliver phase activities feed threat model and assurance case risk measurement. CVSS scores, acceptance criteria, audit results, and assurance confidence levels are example data and metrics for this phase. Acceptable risk determines artifact control gate transition from Release and Deliver to Deploy.

Deploy

Virtual machines and containers are deployed during the Deploy phase. Common activities for this phase are infrastructure provisioning automation, post-deployment checkout, and systems and infrastructure post-deployment security scanning [DoD 2021b].

Data and metrics from Deploy phase activities feed threat model and assurance case risk measurement. CVSS scores, digital signatures, checksums, and audit results are example data and metrics for this phase. Acceptable risk determines artifact control gate transition from Deploy to Operate.

Operate

System and application operations occur during the Operate phase. Activities in this phase include backups, scaling, and load balancing. An operations dashboard provides visual situational awareness of status, alerts, and actions [DoD 2021b].

Data and metrics from Operate phase activities feed threat model and assurance case risk measurement. CVSS scores and various operational incident data that affect confidentiality, integrity, and availability are monitored and measured in this phase. Operational feedback is recorded in the backlog for ongoing development sprints and assurance case development.

Monitor

System, application, and pipeline operations information is collected and assessed during the Monitor phase. Activities in this phase include log aggregation and analysis, continuous monitoring, alerting, asset inventory, security monitoring, and system configuration monitoring [DoD 2021b].

Data and metrics from Monitor phase activities feed threat model and assurance case risk measurement for systems and applications as they are developed and operate. Alerts, scans, and other operational data are continuously measured to provide metrics for risk assessment and determination.

References

[DoD 2021a]
United States Department of Defense. *DoD DevSecOps Fundamentals Version 2.0*. March 2021. https://software.af.mil/wp-content/uploads/2021/05/DoD-Enterprise-DevSecOps-2.0-Fundamentals.pdf.

[DoD 2021b]
United States Department of Defense. *DoD DevSecOps Fundamentals Guidebook: DevSecOps Tools and Activities Version 2.0,* 2021, https://software.af.mil/wp-content/uploads/2021/05/DoD-Enterprise-DevSecOps-2.0-Tools-and-Activities-Guidebook.pdf.

[Ellison 2008]
Ellison, Robert J., Goodenough, John, Weinstock, Charles, Woody, Carol. *Survivability Assurance for System of Systems*. CMU/SEI-2008-TR-008. Software Engineering Institute, Carnegie Mellon University. 2008. https://resources.sei.cmu.edu/asset_files/TechnicalReport/2008_005_001_14978.pdf.

[Ellison 2014]
Ellison, Robert J. *Assuring Software Reliability*. CMU/SEI-2014-SR-008. Software Engineering Institute, Carnegie Mellon University. 2014. https://resources.sei.cmu.edu/asset_files/SpecialReport/2014_003_001_301629.pdf.

[Goodwin 2020]
Goodwin, Mike. *Real-World Threat Modelling*. *Sage*. January 23, 2020. https://medium.com/sagefuturemakers/real-world-threat-modelling-fb14ef767c49.

[Michlin 2017]
Michlin, Irene. *Incremental Threat Modeling*. Presented at Open Web Application Security Project (OWASP) AppSec Europe. May 2017.https://2017.appsec.eu/presos/CISO/Incremental%20Threat%20Modelling%20-%20Irene%20Michlin%20-%20OWASP_AppSec-Eu_2017.pdf.

[NDIA 2008]
National Defense Industrial Association (NDIA) System Assurance Committee. *Engineering for System Assurance 1.0*. NDIA. 2008. https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/sse-committee/systems-assurance-guidebook.ashx?la=en.

[NIST 2021]
National Institute of Standards and Technology. Computer Security Resource Center (CSRC) Glossary. *NIST*.gov Website. September 22, 2021 [accessed]. https://csrc.nist.gov/glossary.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu