# SAIL™: Secure Software, From the Start

NIST observed that "few models explicitly address software security in detail, so secure software development practices usually need to be added to each SDLC [software delivery lifecycle] model."[1] NIST then published a white paper with a set of high-level practices that can be integrated into a secure software development framework (SSDF).

In 2018, Sandia developed a holistic, integrated methodology for a security centric SDLC. This methodology, known as SAIL™ (Software Assurance Integrated Lifecycle) enables us to approach data and software security with a secure system view. Our goal is to identify and mitigate security risks as early as possible and to provide software development teams with actionable information. SAIL™ enables software assurance from two perspectives – as a developer of custom software (that will likely include third party and open-source libraries and frameworks) as well as from an integrator of COTS solutions into broader solutions. We believe that the capabilities and practices we define can be applied to any project, regardless of the existing software development lifecycle, project management methodology, or technology frameworks/programming languages. Software development teams should consider secure software to be a requirement rather than an afterthought: this means developers need a thorough understanding of secure coding practices and know how to get help.

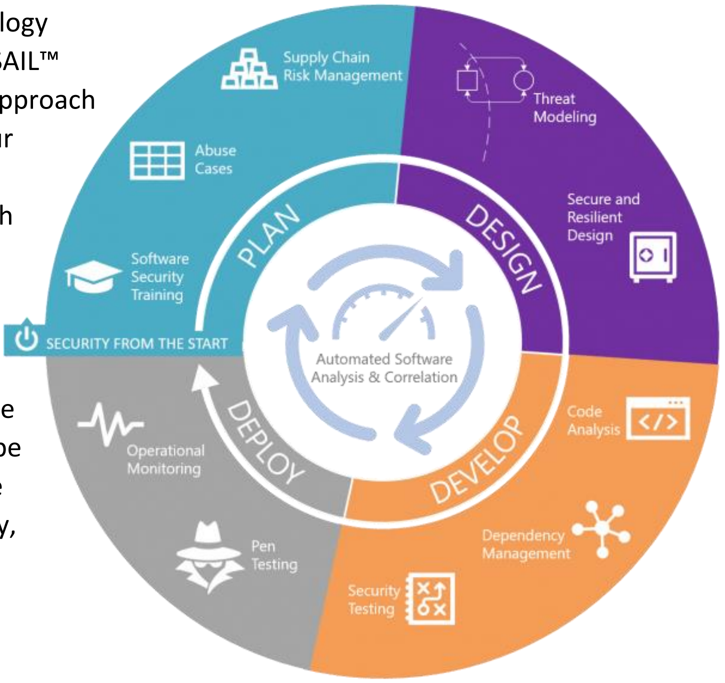

*Figure 1: SAIL(TM)*

## Description of SAIL™ Stages

### Plan

Software Security Training empowers software development teams by raising awareness of the importance of software security, introducing an adversarial mindset, introducing common vulnerabilities, and providing best practices and resources to help developers create more secure software. As Bruce Schneier said, "[W]e should be spending money on security training for developers. These are people who can be taught expertise in a fast-changing environment, and this is a situation where raising the average behavior increases the security of the overall system.[2] Security champions are given hands-on training and deployed within development teams to provide localized security expertise while also creating a network of security-minded professionals.

Abuse Cases help projects plan, prioritize, test, and track security-specific activities. Understanding how use cases can be abused or misused to exploit or harm a system/user helps developers identify security requirements. Developers can also identify potential security controls by designing countermeasures.

Supply Chain Risk Management involves managing exposures, threats, and vulnerabilities throughout the supply chain and developing risk response strategies to the risks presented.

[1] Dodson, et al., "Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)," ii (April 23, 2020), https://doi.org/10.6028/NIST.CSWP.04232020.
[2] https://www.darkreading.com/risk/on-security-awareness-training/d/d-id/1139381

### Design

Threat Modeling is a collaborative and iterative activity with business and technical stakeholders to address security. The process involves identifying worst-case scenarios, flaws in candidate architectures, and potential mitigations. The key to successful threat modeling and security testing is to think with an adversarial or "let's break it" mindset.

Secure and Resilient Design involves designing with security from the start and anticipating the resiliency goals and the areas where security activities may be. Teams are encouraged to use secure design patterns.

### Develop

Code Analysis helps to identify vulnerabilities as software is being developed. We urge teams to use a combination of Static Analysis Security Testing (SAST), Dynamic Analysis Security Testing (DAST), and manual security reviews.

Dependency Management is necessary because although open-source libraries are a known risk, Synopsys found that over 98% of the codebases audited in 2020 contained open-source[3]. Over 60% of organizations surveyed by Snyk[4] do not fully inventory the dependency trees in their software, which can be easily remedied using Software Composition Analysis (SCA) that enables insight into not only open-source dependencies but indirect dependencies, which comprise most open-source vulnerabilities. Ideally, organizations will also manage a repository of trusted libraries.

Security Testing can include manual and automated tests to verify that security mitigations are functioning as expected. Treating security mitigations like business features means security-focused unit, integration, functional, and system-level tests can be developed.  Our expectation is that development teams will be testing early and often rather than relying on a separate test phase. This approach supports various "shift left" and software security efforts that advocate for building security in rather than bolting it on later, which is not only ineffective but costly.

### Deploy

Pen Testing is performed when staging the release of software or in production to identify flaws in environments that real attackers target. If the SAIL™ activities and best practices have been followed, the results of pen testing should be minimal and not include any surprises. Abuse cases can also help prepare for and scope pen testing efforts.

Operational Monitoring includes activities to prioritize and triage security flaws found in the software after deployment. Security activities do not end just because software is deployed. The adversary does not rest, and neither should a development team.

### Correlation and Analytics

Automated Software Analysis & Correlation are at the core of all these activities. Data from the development process and tools can be analyzed to identify continuous improvements. Correlation and analytics also can be used for monitoring the runtime behaviors of software to identify abnormal behaviors and potential security flaws. This capability acts as a check to ensure software is operating as intended, ideally without suspicious/malicious behavior.

## Proposed Path Forward

We propose leveraging the existing work (done to protect the Nuclear Weapons Stockpile and Sandia-developed applications) to create pervasive standards and artifacts that are associated with various stages of the lifecycle. SAIL™ has already been adapted by other organizations to institutionalize a secure SDLC process and provide evidence and understanding to users and stakeholders that appropriate action has been taken to ensure secure software, from the start. Insecure software is everywhere and releasing secure software development lifecycle standards and best practices, such as those referenced here, will be key to improving the security posture of software purchased by the federal government.

[3] https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html
[4] "The State of Open Source Security 2020" https://snyk.io/open-source-security/