Application Security Guide for Software Engineering Leaders

29 November 2024 - ID G00789097 - 12 min read

By: Aaron Lord, Manjunath Bhat

Initiatives:Software Engineering Technologies; Build a World-Class Software Engineering Organization

As cyberattacks and security breaches continue to rise, ad hoc manual security testing will fail to ensure the security of applications. Software engineering leaders must integrate security into the software development life cycle and enable teams to modernize application security with automated testing.

Additional Perspectives

- Summary Translation: Application Security Guide for Software Engineering Leaders (11 July 2023)
- Invest Implications: Application Security Guide for Software Engineering Leaders (21 June 2023)

Overview

Key Findings

- Many software engineers are primarily focused on implementing functional requirements but are lacking the awareness of secure coding practices that prevent security vulnerabilities from being introduced into code.
- Engineering organizations often lack correlation, visibility and orchestration of the software development life cycle (SDLC). Because modern application security requires integrated automation, software engineers face the kind of inconsistencies and friction that make it difficult to scale and mature their security practices.
- As applications gradually become more complex, engineering teams find it increasingly difficult to ensure application security. When security is not part of the application design process, unknown vulnerabilities will emerge over time.
- Engineering organizations still use manual security testing at the end of the SDLC, which leads to delays and frustrations for software engineers.

Recommendations

Software engineering leaders responsible for application security should:

- Educate software engineers about the risk and impact of introducing security vulnerabilities by increasing awareness and implementing active learning of secure code practices.
- Correlate, orchestrate and visualize the SDLC process to promote consistent, frictionless application security by evaluating modern SDLC solutions for security policy orchestration and software delivery data correlation.
- Generate security requirements to reduce the risk of vulnerabilities stemming from design flaws by applying secure design principles and threat modeling during the application design process.
- Implement security guardrails at every stage of the SDLC process to reduce security-related delays and frustrations by embracing automated security testing.

Introduction

The security of applications has become a primary concern for software engineering leaders as the attack surface of their organizations grows larger and more lucrative. Gartner research reveals that more than half of software engineering leaders are directly responsible for application security, and another third share that responsibility. [1] Malicious actors now prioritize attacks against an organization's hosted applications. To counter these attacks, leading engineering organizations are modernizing application security and SDLC processes.

In most organizations, however, application security practices lag behind the curve of innovation. Engineering organizations often do not have a holistic visibility of the SDLC. Even if they do, many software engineering teams continue to rely on manual, piecemeal security processes at the end of the SDLC. The kind of engineers that lack the skills and tools to effectively secure applications will rarely collaborate with the kind of engineers that are most knowledgeable about vulnerabilities in source code.

As a result of all these obstacles, software engineering leaders are unable to integrate application security within DevOps practices. Thus, their teams cannot effectively protect the organization from costly attacks and breaches.

How can software engineering leaders overcome these barriers to improve the security of applications? This research outlines four key actions that you must take to modernize application security (see Figure 1):

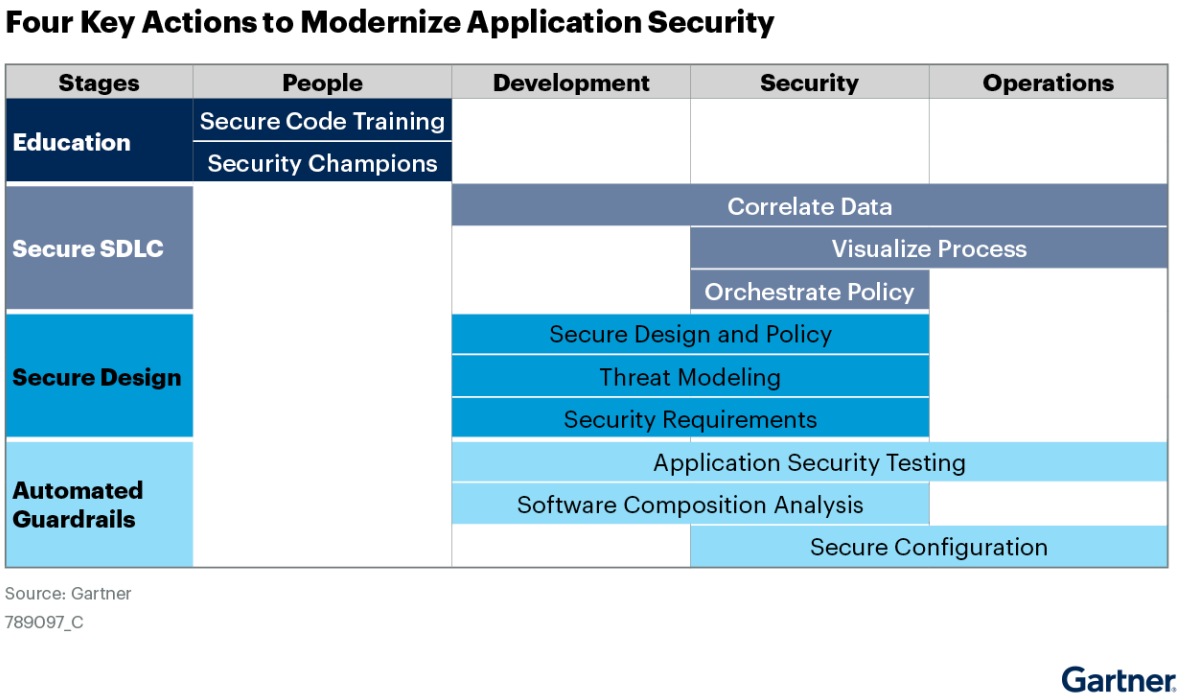
1. Educate developers about the risk and impact of introducing security vulnerabilities

2. Correlate, orchestrate and visualize the SDLC from design to deployment.

3. Generate security requirements during the application design process.

4. Implement security guardrails at every stage of the SDLC and embrace automated security testing.

Figure 1: Four Key Actions to Modernize Application Security



Analysis

Educate Developers About Secure Code Practices

Software engineering staff require specialized training in secure code practices to avoid common security pitfalls. Online learning platforms are most typical and are hosted online for self-service consumption. Commercial services will offer access to training at a subscription cost (see Develop Your Technical Skills Using Online Learning Platforms).

> **In a recent Gartner survey, 75% of software engineering leaders stated that application security skills are "a pain point in their organizations," making this skill set the single greatest obstacle for software engineering leaders.**
>
> *— Source: 2022 Gartner Software Engineering Leaders Role Survey [1]*

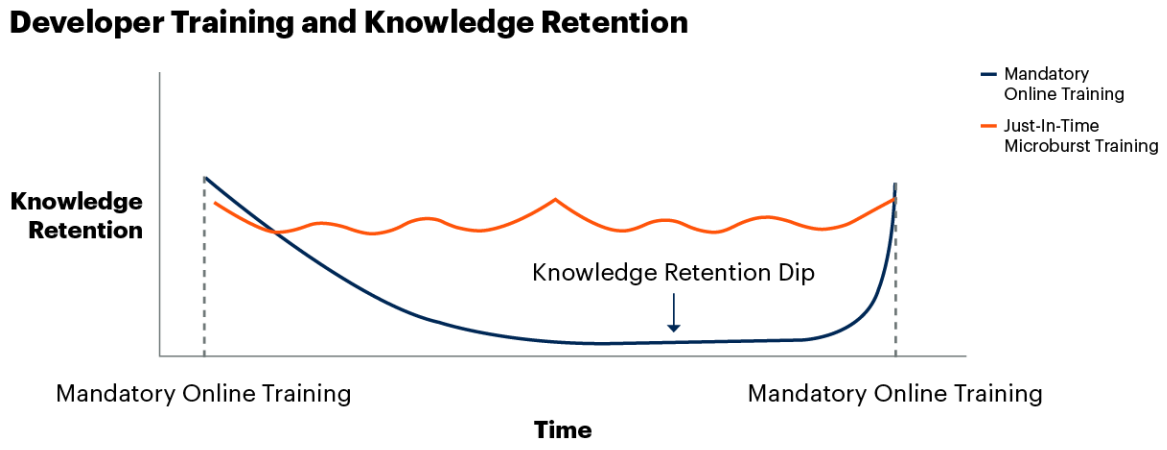
While these types of training should be mandatory, it is not enough. Developers may only use online learning platforms once or twice a year, so it is tough to know if they have retained and applied that knowledge. Also, online learning platforms tend to be technology-agnostic, as it is difficult to tailor every training to every technology stack or coding language. As a result, some security training may not be useful or engaging.

Software engineering leaders should layer real-time embedded training into their SDLCs using automated tools and integrations. For example, results from static application security testing (SAST) and dynamic application security testing (DAST) tools can be contextualized with targeted training that shows how this issue was created, what is the impact of this issue, and how it is remediated.

Deploying training at scale can present additional difficulties. Most engineers do not have the time for full security training on every application, and not every engineer can also be a security professional. Software engineering leaders should promote a "security champions" program to identify developers who have a passion for security. These champions can receive additional application security training, additional responsibilities, and additional incentives. Security champions can act as mentors to scale security awareness in the engineering organization. See Ignition Guide to Designing and Launching a Security Champion Program for details on how to begin.

When developers receive just-in-time microburst training alongside mandatory online training, secure coding practices will be retained over the long term (see Figure 2).

Figure 2: Developer Training and Knowledge Retention



Source: Gartner
789097_C

Gartner

Correlate, Orchestrate and Visualize the SDLC Process

The SDLC is the prescribed process of software creation, from developers designing and writing code, to having it built and hosted in a production environment. Software engineering leaders must ensure that application security testing occurs earlier in the SDLC process and more often. To accomplish this, you must correlate security and software delivery data, orchestrate security policy and visualize the secure SDLC process.

To overcome the lack of visualization and oversight, software engineering leaders must correlate and orchestrate the SDLC and improve visibility.

To deliver SDLC correlation, visibility and orchestration, software engineering leaders should evaluate the following solutions:

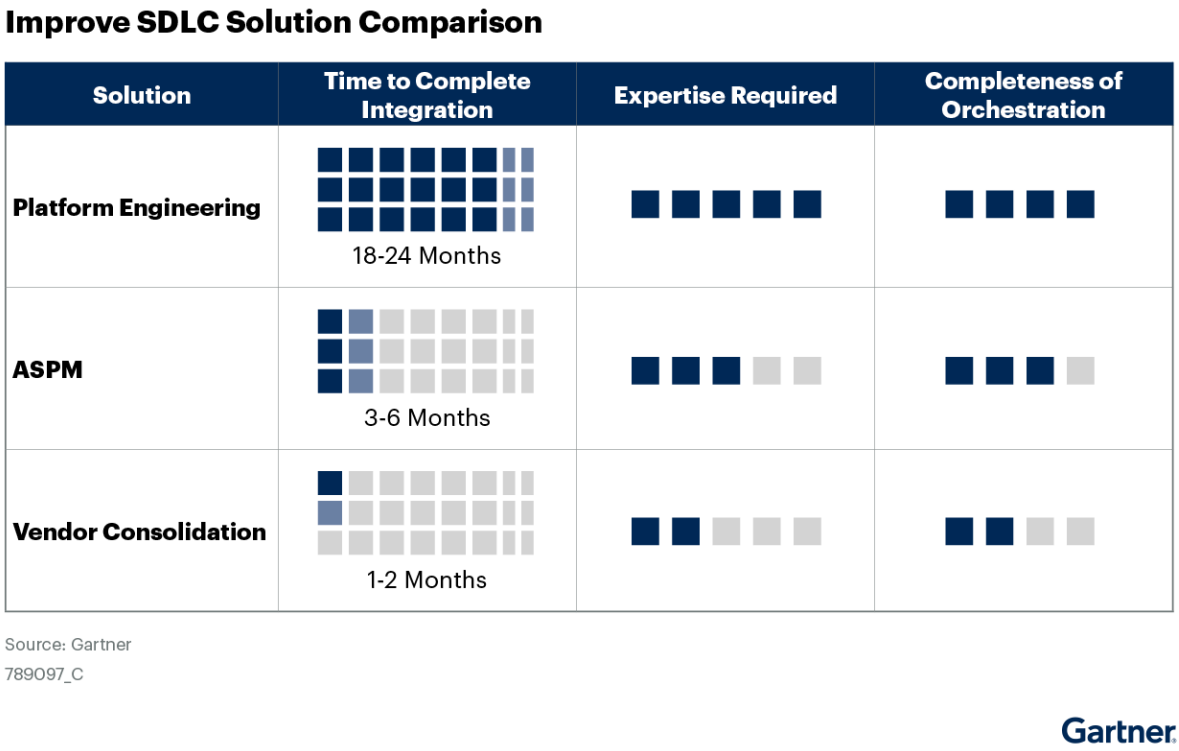
1. **Create a platform engineering team:** Platform engineering is the trending discipline of building and maintaining an internal development platform (IDP). The platform is a layer that is built and owned by an internal platform engineering team that supports the needs of development, security and operations. The goal for a successful IDP is to streamline software delivery and improve developer experience. Software engineering leaders should evaluate if platform engineering is a good option to correlate security and software delivery data, orchestrate security policy and visualize the SDLC (see How to Start and Scale Your Platform Engineering Team).

2. **Use application security posture management (ASPM):** ASPM is an emerging technology that can be used to integrate the SDLC and security tooling. It provides a single-screen view of all correlated application security data. This integration allows engineering teams, security teams and operations teams to orchestrate security policies in ASPM that will automatically be carried out across multiple pipeline and operation tools.

3. **Strategize for cybersecurity vendor consolidation:** Cybersecurity vendor sprawl is a growing problem, causing engineering organizations to see increased inefficiencies and decreased collaboration between silos. Organizations are rapidly pursuing strategies for cybersecurity vendor consolidation. [2] Software engineering leaders should evaluate where cybersecurity tools can be combined into a single platform to improve correlation, visualization and orchestration.

Figure 3 outlines a comparison of the three SDLC modernization solutions. Software engineering leaders should blend these solutions in a way that best fits their organizations. For example, they could consolidate cybersecurity vendors for certain phases of the SDLC and use ASPM for the rest of the cycle.

Figure 3: Comparison of Modern SDLC Solutions



Software development system security is also vital to the SDLC, as attacks on the software supply chain are a growing concern. Protecting the integrity of internal and external code, hardening the software delivery pipeline and governing resource access for software engineers will decrease the threat of malicious code injection (see How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks).

Generate Security Requirements During the Application Design Process

Application security must be a forethought for teams designing new software. A lack of secure design and threat modeling will introduce the most prevalent and difficult vulnerabilities to remediate. Further, a lack of communication between development and security will lead to applications being released to production without secure design verification.

Software engineering teams must work with security to conduct threat modeling to ensure that development is not introducing vulnerabilities that impact the entire codebase, architecture and infrastructure.

Software engineering leaders must apply the following three procedures to generate security requirements during application design:

1. **Architect applications with secure design principles:** Increased complexity in an application over time will be inevitable. More complexity means more bugs, and more security issues. “Secure by design” is a term that encompasses the idea that the default behavior of an application must follow a secure path. Secure by design is a complex topic that is highly dependent on the application itself and its environmental factors. By ensuring that the default behavior of an application is secure, some security bugs introduced later in the application’s life cycle will already be mitigated. See Essential Skills for Security in Modern Application Development for a great starting point for secure design principles.

2. **Implement threat modeling during application design:** Threat modeling is the practice of discovering possible threats that an application may incur. By discovering these threats during application design and planning, engineering teams get a better idea of the security requirements that they need to be concerned with (see 5 Frequently Asked Questions About Threat Modeling). Threat modeling automation tools are emerging that help software engineering and security teams to automate threat modeling.

3. **Verify security requirements with security unit, integration and acceptance testing:** While most engineering teams test applications before release, it is less common to test code for security or abuse cases. Designing these security test cases is difficult, and it can be unclear where to begin, but, security testing is essential. Security bugs and vulnerabilities tend to not follow expected outcomes or adhere to business rules. Generating security requirements with secure design and threat modeling during the design phase will help to define these security test cases. Software engineering leaders should promote layering security tests that will help to ensure consistent verification for threat mitigation.

Implement Security Guardrails in the SDLC

Relying on security teams to verify application security late in the SDLC is not compatible with Agile and continuous integration and continuous delivery (CI/CD) methodologies. It is far more effective to integrate security into the established tools and procedures in use by development. This approach reduces friction with deployment of features and allows for security testing to scale.

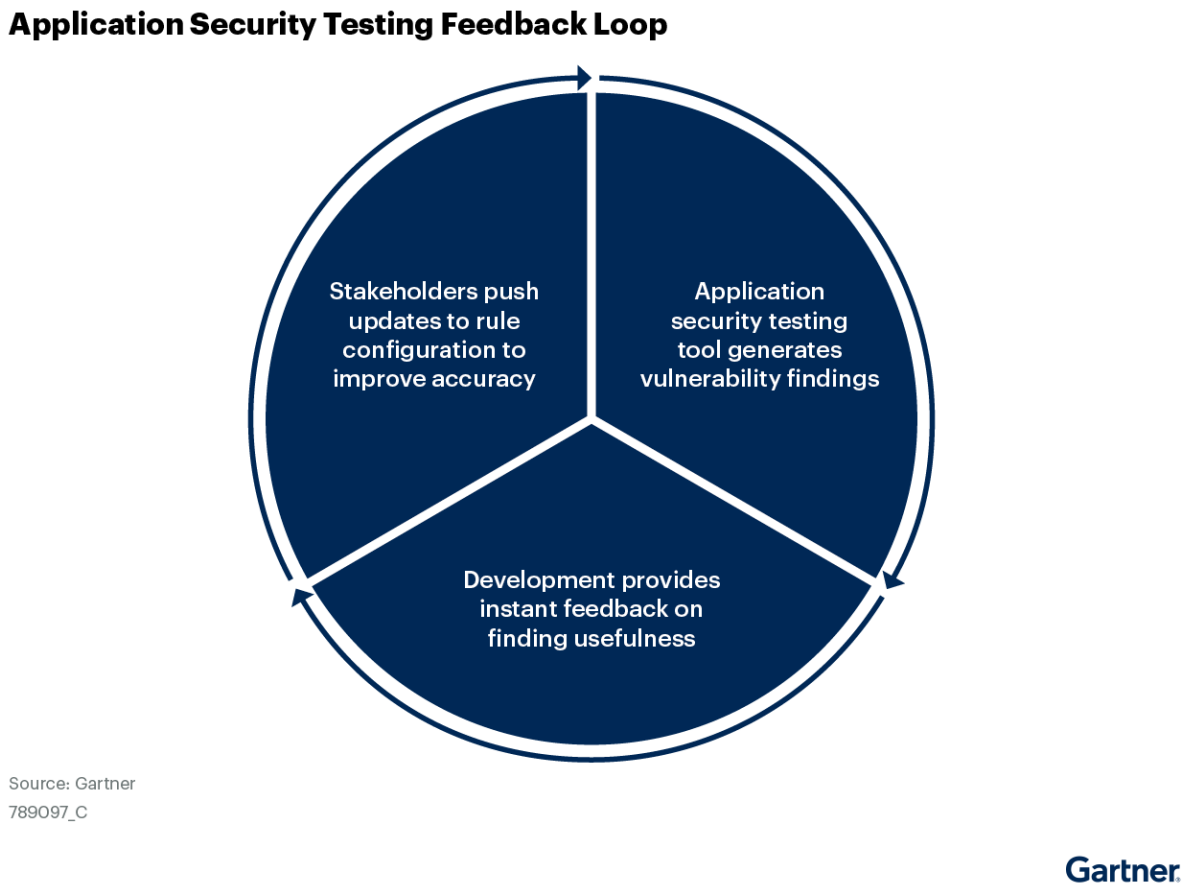
Software engineering leaders must embrace security automation and integrate it into the SDLC. In all phases, they should follow three best practices:

1. **Carefully plan the integration of automated testing:** Integration of new automated tooling requires careful planning, as automated tools will require tuning over time to improve the accuracy of their findings. Integrating automation too quickly without improving accuracy will inundate development with false positive discoveries and vulnerability reports that are difficult to decipher. Software engineering leaders should work with teams that are willing to be early adopters for automation. This will help ease onboarding of new security automation by allowing developers to gradually familiarize themselves with new tools and processes.

2. **Focus on automated tests that provide fast flow and fast feedback:** Application security testing (AST) generally works on a list of rules that determine if a scan has discovered a vulnerability. To improve the accuracy of scanning, software engineering teams must maintain and configure these rules. Once findings are delivered, developers require a way to indicate the helpfulness of these findings quickly. This feedback loop allows for targeted configuration of automation to improve scanning capabilities and accuracy (see Figure 4). Automated tests that take longer than five to ten minutes to run should be completed outside of the SDLC process.

3. **Crowdsource security automation configuration:** To assist in scaling security automation, software engineering leaders must anticipate the need to configure some tool segments using crowdsourcing. This can include software engineers, security champions or coaches, developer experience engineers, and site reliability engineers. This will assist in scaling security automation along with the growth of development and allow stakeholders outside of the security team to have an involvement in their own application security (defining or refining tests for AST, for example).

Figure 4: Application Security Testing Feedback Loop



To establish modern security approaches, software engineering leaders must evaluate key enabling technologies in the application security testing (AST) space. There are multiple categories of AST tools, including:

- **Static application security testing (SAST):** Scans source code for security vulnerabilities.

- **Dynamic application security testing (DAST):** Tests the running application by interacting with its URLs and endpoints.
- **Interactive application security testing (IAST):** Applies a combination of static and dynamic techniques to correlate findings and generate new forms of possible vulnerabilities.
- **Infrastructure-as-code (IaC) security scanning:** Discovers misconfigurations in IaC and flags them for remediation.
- **Software composition analysis (SCA):** Scans open-source and third-party components for known vulnerabilities that are outside of the engineering organization's direct control.

Software engineering leaders must also evaluate additional AST functions that should be implemented into the SDLC, such as:

- API testing and discovery
- Business-critical AST (from SAP, Oracle or Salesforce, for example)
- Container security scanning
- Fuzzing
- Mobile AST (MAST)

See Gartner's Magic Quadrant for Application Security Testing and Critical Capabilities for Application Security Testing for details of vendors and products in the market.

Evidence

[1] **2022 Gartner Software Engineering Leaders Role Survey:** This survey was conducted to understand how organizations attract, hire and retain software engineering talent; improve and modernize developer skills; improve developer productivity; establish platform engineering teams; create platform teams; and incorporate design into software engineering.

The survey was conducted online from November through December 2022. In total, 300 respondents were interviewed from the United States. Qualifying organizations operated in multiple industries (excluding IT software and public sector) and reported enterprise wide revenue for fiscal year 2021 of at least $250 million or equivalent, with 60% over $1 billion in revenue. Qualified participants were highly involved in managing software engineering/application development teams and the activities they perform.

Disclaimer: Results of this survey do not represent global findings or the market as a whole, but reflect the sentiments of the respondents and companies surveyed.

[2] Top Trends in Cybersecurity — Survey Analysis: Cybersecurity Platform Consolidation

Thirty-seven percent of organizations use products from 10 to 20 security vendors and 6% work with products from more than 20 vendors. Many of these products present similar/overlapping capabilities, making it easy for misconfigurations to occur, and difficult to uncover security gaps and to integrate these products.

The goal of security consolidation is to reduce this complexity. By minimizing and streamlining the number of products, perhaps via the use of a broader platform, we can increase security effectiveness.

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

The AI-Era Learning Manifesto: Outcome-Driven Agile Learning

Guide to Application Security Concepts

3 Essential Steps to Enable Security in DevOps

Essential Skills for Security in Software Engineering

Magic Quadrant for Application Security Testing

Critical Capabilities for Application Security Testing