VERACODE

Secure the SDLC in 6 Steps:

Optimizing Developer Experience Through Automation

Table of Contents

Introduction

How can you mitigate potential risks as your attack surface expands with more code streams, frequent releases, and a wider variety of advanced technologies?

The answer lies in automating security in the software development lifecycle (SDLC). This e-book will guide you through the six essential steps to secure the SDLC while optimizing the developer experience through automation. Whether you're just starting with application security testing or looking to scale your existing approach.

This e-book will provide you with valuable insights and modern best practices, including making use of Artificial Intelligence (AI).

The Growing Need for Secure Software Development

Modern software development isn't a linear process. Code is designed, written, built, and tested — and written, built, and tested again and again. Features branch out. Dependencies weave in. Environments intersect. As the volume of code, the number of streams, and the complexity and diversity of environments all continue to grow, teams leverage innovative tools and approaches to keep up with demand.

The demand stems from the need to gain an edge in time to market, facilitate mergers and acquisitions, satisfy increasing regulatory pressures, and minimize operational risk across the software supply chain and widening attack surfaces.

Additionally, organizations are "shifting left." This means they're increasingly reliant on software developers themselves for orchestrating and executing application security testing. These developers are often left with far fewer resources than is appropriate to securely integrate security testing across the SDLC.

Finally, modern software development techniques are creating flaws faster than they can be fixed. When using tools like third-party libraries, microservices, code generators, and large language models (LLMs), we can code faster and more easily. However, research shows that these tools also make it easier to write code that's insecure. As a result, organizations are left vulnerable to potential cyber attacks and data breaches.

The Opportunity: Securing the SDLC

Integrating security into the SDLC, not just as an afterthought or post-solution addition, ensures that proper procedures are in place to create safe digital products quickly, build business resilience, and reduce risk across your entire software supply chain.

By adding security into each step of the SDLC, the most important outputs of the SDLC are assured to be secure when deployed and attestable for compliance. Integrating security into each DevOps function effectively creates DevSecOps – an overarching layer covering all activity along the SDLC.

Furthermore, integrating security into the SDLC can greatly improve the developer experience. With security being a top priority from the beginning of the development process, developers are equipped to focus on building high-quality, functional code. This not only increases efficiency and productivity, but it also allows developers to take pride in their work and feel confident in the security of the end product. Additionally, they gain valuable knowledge and skills that'll benefit them in their career growth and make them more valuable to the organization.

In the following pages, we'll detail six steps to integrate security into your SDLC—with examples of how others have done so successfully. Although the steps might not occur in the same order for everyone, they form the six essential elements we've consistently seen helping organizations secure their SDLC.

Step 1:
Discover and Assess Risks

First things first, before jumping into scans and tooling and integrations, take a step back. The first important step in securing your SDLC is to figure out what applications you have and where the risks are. This will help you decide what to focus on.

Consider asking: How many applications do you have? Where do they reside? Who owns them? Are they still around? What are your open-source dependencies? Is AI being used to generate code?

Understanding these systems, their owners, and the inventory of applications will allow you to include the appropriate stakeholders and integration points throughout the initiative. It's important to understand how different systems and tools are used for legacy versus cloud-native applications, as well as where outsourced and acquired applications fit into your broader portfolio.

Keep an eye out for the following when conducting discovery:

Different Application Types

Identify the entire portfolio, including micro-services and legacy, cloud-native, web, mobile, outsourced, and acquired apps.

Different Application Elements

Establish a complete inventory of applications, the first-party code, their frameworks, additional third-party components, and their deployment artifacts.

Different Development Tools and Ecosystems

Discover which integrated developer environments (IDE), external dependencies, developer tools (including AI), languages, and frameworks are used or supported. You also want to know specific CI/CD pipeline systems and cloud environments which help map out Code, Build, and Deploy aspects of the SDLC (ex: Github Actions, Gitlab Pipelines, Azure Pipelines, Jenkins Pipelines).

Different Risk Levels

Understand which assets in your inventory carry the most risk (likelihood and impact) and have the most precious or valuable data. Consider threat modeling, but balance deep threat modeling exercises with understanding current risk state.

Step 2:
Establish Prevention Methods

Once a thorough inventory of all application elements and development tools has been created, the next step is to establish prevention methods. Prevention is the most effective way to secure your SDLC. By establishing security controls in the development process, fostering a secure coding culture, and providing developers with the necessary tools and guidance, you can prevent security flaws from reaching production.

Testing Tools:

Consider which testing tools and techniques are most appropriate for your organization, and establish how you'll incorporate them into your development process.

A few tools that are easily integrated into the SDLC are:

Static Application Security Testing (SAST)

Scanning source or binary code in a pre-production or "static" state to find security flaws or CWEs. Also called "white box" testing.

Container Security

Scanning to find Container and/or Infrastructure as Code (IaC) configuration risks, secrets exposure, and misconfigurations you can remediate.

Software Composition Analysis (SCA)

Analyzing the open-source libraries and third-party components in an application for CVEs and other known vulnerabilities. Used in the creation of a Software Bill of Materials (SBOM) and supports standard formats: CycloneDX and SPDX.

A word to the wise:

Claims about SAST can be misleading. Some vendors claim to do SAST, but they're primarily looking at IaC configurations and secrets. You'll notice in the description that SCA isn't the same thing as testing first-party code. Many vendors don't even test source code or application binaries themselves. That is why, at Veracode, we pride ourselves on being an organization that continues to research and add coverage beyond expectations. Veracode scans binaries and has support for nearly 40 languages and hundreds of frameworks.

AI-assisted Remediation

Secure-by-design AI-assisted remediation is a critical prevention tool that makes for fast and accurate vulnerability remediation. Leveraging a GPT-based machine learning model trained on Veracode's proprietary dataset, Veracode Fix is a specialized AI that excels at fixing insecure code and dramatically reduces the work and time needed to remediate flaws.

This tool makes two critical pieces of securing software possible. First, when combined with scanning in the IDE, developers can find and fix flaws before the feature branch development is merged back to the main. Second, it makes it possible to tackle massive amounts of Fix-able security debt that would not otherwise be addressed (which we'll cover in Step 5).

Developer Education

Additionally, investing in ongoing developer education and training can help increase awareness and understanding of potential security vulnerabilities, further strengthening your prevention efforts. The State of Software Security 2024 shows us that organizations using Veracode's Security Labs immersive developer education tool have 37% security debt. Compare that to 48% among application teams that don't. The time-to-fix difference is even more significant. Applications developed by teams that aren't using the Labs take seven months longer to reach that 37% mark.

01

Step 3:
Onboard and Scale

Once you have a clear understanding of your application portfolio and methods for flaw prevention, you can onboard applications and conduct initial scans to establish a baseline and gain visibility into your security posture.

Automation is key to optimizing the developer experience and ensuring security throughout the SDLC. By onboarding and scanning applications via integration and automation, you are laying the foundation for a more secure — and cost effective — application lifecycle.

Automating security controls and scanning via SAST and SCA in the IDE, means you can minimize manual effort and reduce the risk of human error. Automation also enables developers to focus on their core tasks while ensuring that security is integrated seamlessly into the development process.

To establish a strong security baseline, it's essential to implement continuous scanning throughout the SDLC. This involves automating security scans for both legacy and cloud-native applications. By integrating security tools into your development process, you can ensure that scans are performed regularly and consistently. Continuous scanning not only helps identify vulnerabilities but also provides developers with real-time feedback, enabling them to address security issues promptly.

Step 4: Set Policies

To align security and development teams, it's important to define and document security policies for your applications. These policies should consider factors such as risk tolerance, regulatory requirements, and application criticality. Different applications may have different policies based on their importance and operating environment.
Policy is a mechanism for risk and compliance objectives to be communicated effectively to development. Without it, you're unable to manage risk and compliance because there is no signal between the roles.

Having a common policy sets clear expectations and rules with which applications must comply. It's a complex task that should include security, development, and even board-level input on qualifying risk tolerance so a policy can be set.

Via technical controls in the CI/CD process, you can establish a continuous feedback loop that monitors compliance with policies and ensures that if code drifts from policy – or if any new flaws are created – teams are alerted, and the violations can be addressed. When scanning with Veracode, you can use Veracode's built-in security policies recommendations for applications based on business criticality. You can also tailor them and restrict findings by severity, CWE category, CWE ID, license risk, Common Vulnerability Scoring System (CVSS) score, or a common standard such as OWASP, OWASP Mobile, CWE Top 25, or Payment Card Industry Security Standards Council.

Since Veracode is built in an auto-scaling cloud environment, Veracode always ensures that our customers have all the computation resources needed to perform testing effectively and comprehensively. Veracode fully tests applications, giving a full view of risk. Policy becomes the language between security and development

Step 5:
Prioritize and Address Findings

Once applications are onboarded and scanned and policies are set, it's time to prioritize and address security findings. This involves categorizing and resolving policy-violating flaws through remediation or mitigation. Remediation involves actively fixing vulnerabilities, while mitigation involves implementing and documenting controls to reduce risk. Research shows that roughly 63% of applications have flaws in first-party code and 70% contain flaws in third-party code. That's why testing both throughout the SDLC is so critical. More concerning still, almost half (46%) of organizations have persistent, high-severity flaws that constitute critical security debt.

If you haven't been integrating security into development thus far, you're likely going to fall into that group of organizations with critical security debt. That's why we covered prevention first. Research reinforces the importance of speed in flaw remediation. Development teams that address flaws promptly can reduce critical security debt by a remarkable 75%. By fixing vulnerabilities quickly, these teams build habits and muscle memory around fixing security flaws, significantly enhancing their security posture and reducing the prevalence of security debt in their applications.

Security and development teams must work together to make sure findings are resolved efficiently. AI-assisted remediation tools can help with addressing findings by rapidly reducing time spent on a tech debt burn-down campaign. Also, regular communication and collaboration between teams is needed to ensure findings are properly understood and addressed.

Step 6:
Leverage Reporting and Analytics

Reporting and analytics provide valuable insights into the effectiveness of your security measures and help track progress over time. A platform's unified reporting system allows you to make sense of data from different tools and systems. Using a platform with robust analytics, you can have reports available on-demand of the real-time status and health of your strategic, measurable application security program. Utilize these reports to create a starting point, pinpoint areas for enhancement, establish measurable objectives, and monitor progress towards achieving those objectives. By leveraging comprehensive reporting tools, you'll gain visibility into your security posture and identify areas for improvement. These reports can be used to demonstrate compliance with regulatory requirements and provide assurance to stakeholders. Analytics can also help identify trends and patterns, enabling proactive security measures.

> **From a security management perspective, the visibility into our software and progress being made through automation are paramount to me for reports to the board. Veracode, as a central tool for our visibility and vulnerability management, is very helpful. I use the reports to establish a baseline, identify areas for improvement, set quantitative goals, and track progress against those goals.**
>
> -Nils Brenneis, Information Security Manager, HDI Global SE

Pro Tip:

Asking the right questions results in measuring the right things. You can start by asking questions such as: How many flaws do we have? How quickly are we fixing them? Who is shipping the safest/riskiest code? How do we compare against similar organizations in the industry?

Conclusion

By integrating security into each step of the SDLC, organizations can create a culture of security awareness, reduce risk, and ensure the development of secure and compliant software.

The six essential steps outlined in this e-book provide a roadmap for optimizing the developer experience through automation while mitigating potential risks. From discovering and assessing risks to leveraging reporting and analytics, each step plays a vital role in securing the SDLC.

By establishing prevention methods, such as using testing tools, AI-assisted remediation, and developer education, organizations can prevent security flaws from reaching production. Onboarding and scaling applications through automation further enhance security and cost-effectiveness. Setting policies and prioritizing and addressing findings ensure that security is integrated seamlessly into the development process. Collaboration between security and development teams is crucial in resolving security flaws promptly and reducing critical security debt.

Finally, leveraging reporting and analytics provides valuable insights into the effectiveness of security measures and helps track progress over time. By asking the right questions and using comprehensive reporting tools, organizations can gain visibility into their security posture, identify areas for improvement, and demonstrate compliance to stakeholders.

Securing the SDLC is an ongoing process that requires continuous effort and adaptation. By following these six steps and embracing a culture of security, organizations can build resilience, protect sensitive data, and stay ahead of potential cyber threats.

Schedule a demo of Veracode and a security pro will gladly help you implement these steps in your organization.

About Veracode

Veracode is a global leader in Application Risk Management for the AI era. Powered by trillions of lines of code scans and the proprietary AI-generated remediation engine, the Veracode platform is trusted by organizations worldwide to build and maintain secure software from code creation to cloud deployment. Thousands of the world's leading development and security teams use Veracode every second of every day to get accurate, actionable visibility of exploitable risk, achieve real-time vulnerability remediation, and reduce their security debt at scale. Veracode is a multi-award-winning company offering capabilities to secure the entire software development life cycle, including Veracode Fix, Static Analysis, Dynamic Analysis, Software Composition Analysis, Container Security, Application Security Posture Management, and Penetration Testing.

Learn more at www.veracode.com, on the Veracode blog, and on LinkedIn and Twitter.