SECURITY COMPASS WHITEPAPER

The Developer-First Application Security Approach

Security Compass

Taking control of the development process

The move by organizations to DevOps and other rapid development and deployment methodologies has challenged security. The old model of a separate application security team with exclusive responsibility to identify and prioritize vulnerabilities simply does not work in today's environment. The need to release code quickly — and securely — requires a different approach.

"Shifting left" is a good goal. However, scarce security resources and the bottlenecks created by security testing are not compatible with the need for rapid time to market. Developers must take the lead for security in this new approach.

Do you rely completely on application security testing?

Using Application Security Testing (AST) methods like static analysis, dynamic analysis, and penetration testing as the primary way to ensure security worked when software was updated three or four times each year. Organizations had more time between releases than today and secure coding standards were less well understood. The tools required security experts to run and interpret the results, but could identify common coding errors that led to vulnerabilities.

Security testing results were "noisy," requiring teams to scrub false positives and minor issues, then prioritize the vulnerabilities by criticality. In turn, the development teams would refactor and resubmit the code for additional testing. Since the tools were often used late in the development process, remediating the issues required more time. Release dates would often slip due to the scans, and friction between development and security was common.

Security testing is not enough to prevent vulnerabilities

Of course, the root cause of delays was that vulnerability scanners were viewed as the first-line for ensuring security. Development teams are given functional specification describing what the software should "do." That it should be secure was assumed, even though that could only be articulated as "having no vulnerabilities." Security requirements such as "don't accept special characters in user names," "do not hard code credentials," and "allow a maximum of three login attempts" did not exist.

Without anticipating security issues and assigning those controls during the development process, organizations are forced to continue to use testing as the primary means of improving security. The result will continue to be a high number of issues late in the development process, leaving organizations with the choice of delaying releases or releasing software with vulnerabilities.

How can developers ensure application security?

The answer is to provide development teams with the information and tools needed to build security into the code — as they write the code. There are several strategies for accomplishing this, each with its own challenges.

Secure coding standards are not actionable

The most common strategy for developer-driven security is to develop and deploy secure coding standards. In their first iterations, these may cover basic security hygiene, like always use the principle of least privilege or adopt zero trust design strategies, and validate all input to ensure appropriate input type and content. If written standards are required, organizations can adopt existing guidelines from OWASP, CERT, and others.

Secure coding standards are a good start. However, they can be challenging to communicate, follow, and enforce. Developers are accustomed to delivering a fixed set of features by a specific date. More importantly, teams are almost always measured by this standard. When building a feature for release, it may be difficult to remember that policy dictates one must *"Contextually output encode all data returned to the client that originated outside the application's trust boundary."* Beyond remembering the need for this policy, the developer also needs to understand the application's trust boundary and any organizational standards specific to this policy.

In short, while secure coding policies are necessary, they are typically not actionable. A written policy does not provide cues to developers for when a policy is triggered. Examples for developers may exist in a manual, but this can slow progress.

Traditional threat modeling is time-consuming

Threat modeling can address some of the operational challenges with secure coding policies. In a traditional threat model, software architects, developers, and security experts analyze an application's design to identify weaknesses an attacker could exploit. In highly critical environments, teams may even examine the tools, techniques, and procedures used by a specific adversary described in MITRE's ATT&CK Framework.

The output of a threat model is a list of all identified threats and a corresponding set of controls to mitigate the risk that are assigned to developers and QA personnel. Ideally, the threat model also incorporates secure coding policies for the application under review. This effort makes secure coding requirements more actionable, but poses other challenges:

Scarce resources: Only a few organizations have all the security resources they desire. And traditional threat models require weeks of effort from experts to discuss architecture, complete questionnaires, produce data flow diagrams, and select controls.

Scale: Traditional, manual threat modeling does not scale. Enumerating threats and corresponding controls can take weeks. Diagraming architecture and generating attack trees and DFDs require days of discussion.

Consistency: The participants in a threat model are responsible for identifying and ranking threats then prescribing controls. The collective expertise and experience of the team keeps changing as people move on to different organizations. The result can be inconsistent evaluations and controls.

Auditability: Shared spreadsheets or documents can guide development and security teams but are poor choices for providing evidence of compliance in the event of an incident or audit.

Establishing practices to drive enhanced application security

The problem, of course, is not with policies and threat modeling. These both provide well-established methods for building more secure software. Policies call out organizational best practices, so there is consistency in how controls are used across multiple projects. Threat modeling helps teams anticipate threats and weaknesses and assign controls to mitigate risk.

Further, most of these techniques to build more secure code are well known. Even without formal policies, development teams realize they should validate untrusted input, use authentication controls, and encrypt sensitive data. The key is to ensure that that information is provided to developers consistently, as needed, and in a usable form. This requires:

- ***Consistent classification of applications:*** Risk ranking helps organizations focus on which applications are most critical to their business goals. In turn, this guides teams in determining security policies and compliance and privacy requirements for the project. This could include secure coding standards, prescriptive activities, and controls required by standards such as the PCI-DSS, or more general standards of "reasonable security" controls as used in Section 5 of the FTC Act.

- ***Characterize the application's technical stack and associated threats:*** Manual threat models are warranted for an organization's most critical applications. However, up to 90 percent of the applications' threats are a function of the programming language, frameworks, and other aspects of its technical stack. Using an automated threat modeling tool allows organizations to scale threat modeling across all applications and ensures consistency between projects.

- ***Enumerate required controls for each identified threat:*** The key to implementing better security is translating threats, policies, and regulatory requirements into actionable tasks and controls. Rather than generic policies, security controls are specific tasks that mitigate the risk for each identified threat. Each task is assigned to software developers and for coding requirements, QA for test plans, or operational security for server configurations or web application firewall rules. Using automated threat modeling tools ensures consistency and compliance with all internal and external requirements.

Aligning security testing with proactive security practices

By analyzing the technical stack and deployment environment of an application, development and security teams can identify and anticipate up to 90 percent of the threats and weaknesses. Doing so with tools allows organizations to scale threat modeling across a large percentage of their application inventory and ensure consistency of controls.

When threats and regulatory requirements are translated into actionable tasks, development teams can use the information and tools needed to “build security in.” First-order identification of potential vulnerabilities becomes a function of meeting expressed requirements, and security testing becomes a validation exercise to ensure controls were implemented correctly.

As development and security teams gain a common understanding of requirements and controls, it allows organizations to avoid vulnerabilities late in the development process.

Security Compass

Go Fast. Stay Safe.

Security Compass, a leading provider of cybersecurity solutions and advisory services, enables organizations to adopt Balanced Development Automation for rapid and secure application development. With their flagship product, SD Elements, the company helps automate significant portions of proactive manual processes for security and compliance that improves time to market for new technology. In addition, they offer advisory services on howorganizations can embrace emerging technologies like cloud to strengthen their security posture. Security Compass is the trusted solution provider to leading financial organizations, technology enablers, and renowned global brands. The company is headquartered in Toronto, with offices in the U.S. and India. Follow Security Compass on Twitter @securitycompass or visit them at securitycompass.com to learn more.

1.888.777.2211
info@securitycompass.com
www.securitycompass.com

@SECURITYCOMPASS
SECURITY COMPASS

OFFICES

GLOBAL HEADQUARTERS
1 Yonge Street
Suite 1801
Toronto, Ontario
Canada M5E 1W7

TORONTO
390 Queens Quay W
2nd Floor
Toronto, Ontario
Canada M5V 3A6

NEW JERSEY
621 Shrewsbury Avenue
Suite 215
Shrewsbury, New Jersey
USA 07702

CALIFORNIA
600 California Street
San Francisco, California
USA 94108

INDIA
#4.07
4th Floor, Statesman House
Barakhamba Road, New Delhi
India 110001