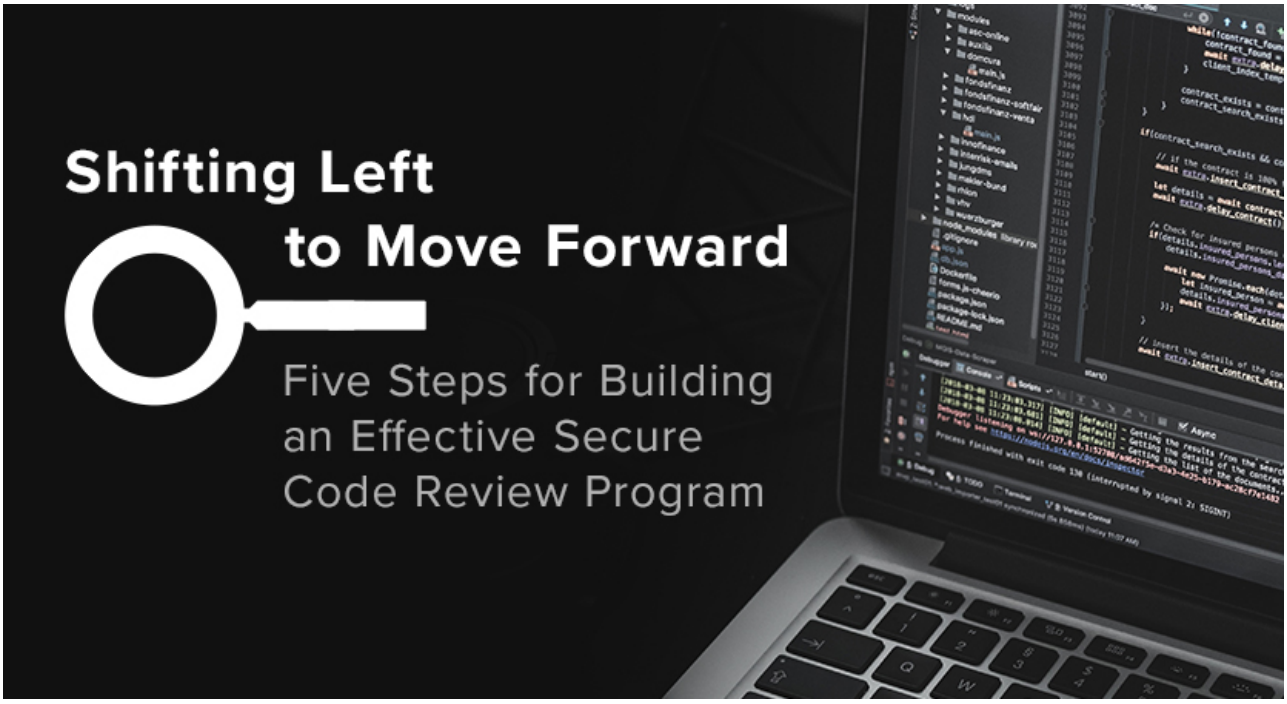
Shifting Left to Move Forward: Five Steps for Building an Effective Secure Code Review Program

written by Paroksh Sharma | November 10, 2020



Today, nearly every company is a software company, resulting in an unbelievable amount of code that's subject to security issues. What's more, a myriad of methods for identifying vulnerabilities focus heavily on a post-deployment approach, leaving security gaps between development and testing.

It's time to shift left.

By shifting left and introducing secure code review (SCR) the moment the first line of code is written (or as soon in the software development lifecycle as possible), any organization creating applications can identify real vulnerabilities well in advance of deployment, thereby increasing team productivity and thwarting future outside attacks, all while decreasing costs of testing for vulnerabilities too late in the software

development life cycle (SDLC). SCR coupled with pentest and threat modelling engagements provides the most in-depth application security testing coverage. It is an essential strategy to any application security strategy. To learn more, read 3 Steps to Reimagine Your AppSec Program in *Cyber Defense Magazine*.

If not done right, failed SCR programs can cause organizations to lose valuable time, money, and effort. It's important to develop a SCR program that offers risk context, uses the right tools, has set processes and standards, and doesn't overwhelm application teams with false positives. Ultimately, the challenge lies in the fact that creating and running a SCR program is not straight forward and one strategy may not fit all organizations. It requires ongoing planning, discussion and possibly, organizational changes. To help, we've compiled five steps to get you started on the right path.

Step 1: Develop a Security Culture and Listen to Your Developers

SCR program planning should not be done by security people alone – be sure to include your experienced developers and application teams when discussing strategies for selecting the right platform, integrating tools in the development ecosystem, and assessing and improving the process.

A solid security culture also aligns code review activity toward assisting developers in writing secure code rather than appearing as an additional burden that delays the release within an already restricted timeline. In other words, look at secure code review as a way to empower developers to write secure code from the start.

Another important element to developing a strong security culture is to rotate code reviewers. This ensure that your source code is regularly reviewed with a fresh pair of eyes and secure code reviewers get exposure to different

development environments.

For enterprise organizations, hold awareness sessions during which various teams share common mistakes or top findings – without calling out the developers. And, when developers do write highly secure code, encourage them to continue to do so with rewards that they find meaningful.

Step 2: Create Simple and Effective Methodology and Processes

Lack of transparency and simplicity in a SCR program can lead to loss of time and frustration, which is why streamlined, clearly documented processes, policies, and guidelines are critical for success. Information that should be made available to all teams include items such as defining expectations, scan frequencies, and how to approach remediation (see below). Above all, keep it simple.

One key to keeping it simple is automation, which should be leveraged to run scans and to track issues and remediation progress. However, customizing off-the-shelf tools is equally as important. But don't rely on tools solely – while automated tools are useful in finding vulnerabilities in less time, there will always be a certain category of issues that they won't identify Make an effort to contextualize your code review process by understanding the application's use case and underlying framework issues. Conducting manual secure code reviews is essential to finding those hidden culprits, particularly in critical applications.

Step 3: Plan Application Onboarding and Scan Frequency

When you're implementing a SCR program, you need to consider multiple factors. Whether you have a large or small inventory of applications, a risk-based approach to scope and static analysis is essential. In other words, prioritize your "crown jewels," business critical applications or those with sensitive data. Scan them more closely and frequently than internal applications that do not contain critical data.

In terms of scan frequency, many organizations adhere to "once-a-year minimum" compliance guidelines to scan all applications or at few major releases. This approach does not work for all applications within a portfolio, however. Instead, leverage static analysis automation tools as frequently as possible in the CI/CD pipeline.

If your organization is moving to an agile environment, where you do development work in sprints, another approach to application onboarding is to implement a separate code review sprint. We all know how much pressure sprints can place on teams to create new code within a constrained timeframe. By dedicating a security sprint that's separate from the development sprint, you can achieve secure code without delay and within specified deadlines. If you decide to take this approach, we recommend implementing a lightweight secure code

review during pre-commit, when new code is reviewed before it is introduced into the code base, to eliminate the introduction of security issues early in the cycle.

Step 4: Understand That Remediation Matters Most

Code reviews are great, but don’t stop there. Once you have your list of vulnerabilities, make sure you also have a plan to remediate them, and enable your developers to do so properly with remediation libraries and guidelines at their fingertips. Include:

- Tools that identify security issues and give feedback as developers are coding
- Readouts after every assessment, thereby giving developers time to examine identified issues and raise questions
- Clear deadlines for fixing problems
- Training to enable developers write secure code

Step 5: Measure and Improve

What you don’t measure, you can’t improve. If you want your secure code reviews to improve over time, then measure and track your progress (or lack thereof). Determine what your key metrics are and find opportunities to gather the appropriate data. Ask yourself questions like:

- What is your remediation rate?
- How much time is remediation taking? Is it increasing or decreasing?
- Are you discovering the same type of security issues and what action should be taken to avoid that ?
- Are developers learning anything from your vulnerability

reports? That is, are they writing better code or are they reintroducing the same issues over and over?

Examine your organization’s performance against your key metrics on a regular basis – quarterly or yearly. Additionally, look beyond code review activity within your own organization to what others are doing. Where do you stand amongst your peers? What are they doing differently?

The ultimate goal of any SCR program is to reduce the time, cost, and effort spent on identifying and fixing security issues that are captured later in the SDLC. The more effective your SCR program is, the less time and money will be spent on analyzing and remediating issues after an application has been deployed.