Checkmarx

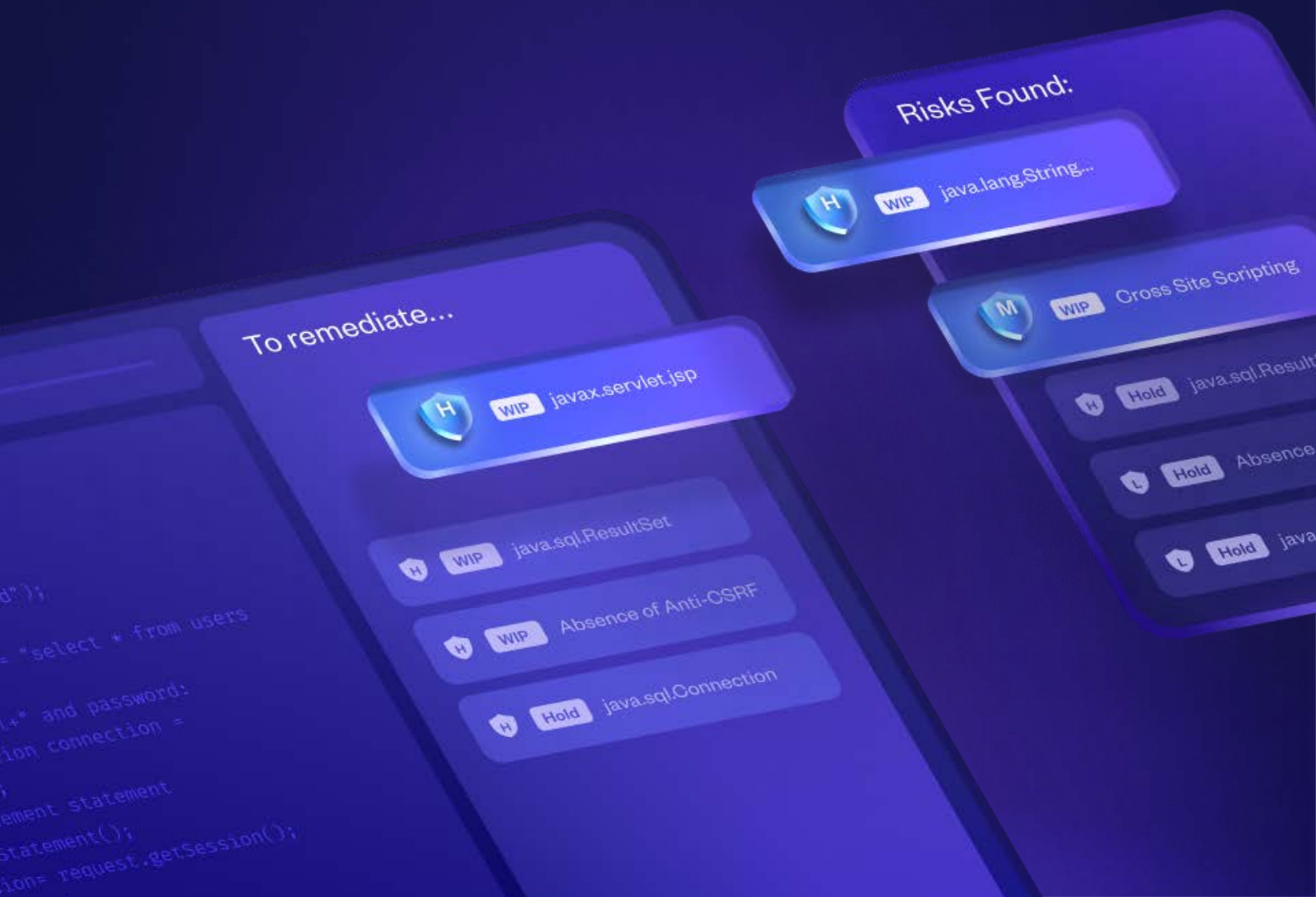
Whitepaper

7 Proven Strategies

to Unlock Developer Adoption

An AppSec team's success is not only measured by its ability to find vulnerabilities, but equally, if not more important, by its ability to get them fixed effectively. Getting strong developer buy-in is an inherent part of it.

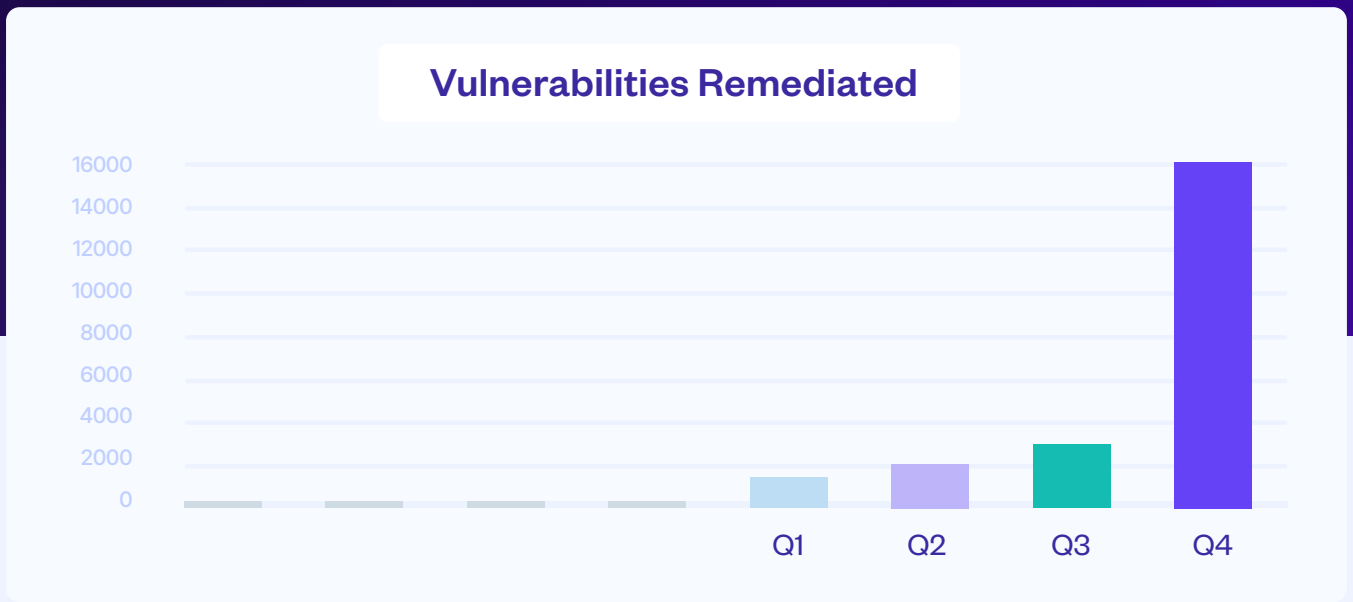
In this guide, we present seven proven strategies to achieve this through cultivating a stronger relationship with dev teams and facilitating adoption.



Getting Vulnerabilities Fixed Faster

The strategies set forth in this document, which are designed to get dev teams to adopt a security-minded approach and integrate security into their workflows through tools, training, and support, don't just look good on paper. They bring tangible results, such as in the case of this Fortune 500 company.

Before following our strategies, they spent the first year and a half remediating only 50 vulnerabilities per quarter – 300 vulnerabilities in total. Afterward, they were able to fix 2,000 vulnerabilities the following quarter and accelerated to a run-rate of over 15,000 vulnerabilities fixed every quarter only six months later.



A leading Fortune 500 company unlocked developer adoption and fixed

149,900% more vulnerabilities

01

Integrate with existing tools and workflows

The allocation of responsibilities is clear: Security finds vulnerabilities, developers fix them. To make that happen, security teams need to make it easier for developers to fix the vulnerabilities they find. That requires integrating into existing tools and workflows to minimize the amount of additional friction. Unfortunately, for many organizations that is not the case: Many organizations still deliver scan results to developers in spreadsheets and PDF reports.

A better way is to provide security findings within the tools that are part of their developers' daily workflow: the IDEs where they write code, source code management (SCM) where they commit and review code changes, bug ticketing systems where they respond to software issues, and collaboration tools where they communicate with other team members. For developers, working on security issues should feel no different than working on feature development or bug fix.

By allowing security tools to integrate directly into the developer's workflow—whether it's their IDE or SCM—developers can stay in the environment they're already comfortable with, ensuring that security becomes part of their day-to-day tasks rather than an external, disruptive process.

02

Point to the Most Critical Fixes

Most developers have more bugs than they can fix, even before you pile vulnerabilities on top. Security teams must think realistically about which vulnerabilities they absolutely need fixed, and which would be merely nice to fix.

It's well known that traditional severity ratings don't reflect real-world risk. Security teams need better ways to assess the risk of a vulnerability. For example, static reachability analysis helps understand if a vulnerability is actually exploitable, while runtime reachability analysis helps understand if a vulnerability is exposed to the open Internet from within a production environment.

Understanding real-world risk and directing developers to only the most critical vulnerabilities to fix can help make a meaningful impact on reducing actual (as opposed to theoretical) risk without overburdening development teams.

03

Help Developers Fix Vulnerabilities Faster

While there's a time and place for training courses, most developers just want to fix what's in front of them at the moment. Security teams can improve DevSec cooperation and trust by making it easier for developers to fix a vulnerability compared to a normal bug: For most bugs, developers need to reproduce the issue before they can diagnose the root cause, identify where in the codebase it lies, and make the fix – often a tedious and time-consuming process.

Similarly, traditional AppSec tools often force developers to leave their familiar environments and navigate unfamiliar security platforms, resulting in frustration and disengagement.

Instead, look for a code security tool that understands the data flow of an application and can identify and point to the exact line of code to fix a vulnerability. Advanced AppSec tools can also include AI guidance or even auto-remediation with code snippets that fix the vulnerability. This speeds up remediation while helping developers learn about the vulnerability, aided by AI assistants that can answer any additional security questions live. By providing in-line guidance and easy-to-understand fixes, developers can address vulnerabilities directly in their development environment, reducing frustration and streamlining their process.

04

Shift as Far Left as Possible

Vulnerabilities will always exist and helping developers fix them faster will always be a necessity. However, empowering developers to address security issues before they enter the codebase can help minimize the amount of technical debt that builds up over time.

Look for tools that allow developers to trigger security scans right from their IDE, so they can validate that their code is secure moving forward. Likewise, the ability to scan local branches allows them to scan code on their machines before it's checked into the repo. New AI secure coding assistants take this a step further by automatically scan code as it's being written, providing immediate feedback to correct issues while developers are in their IDE.

05

Provide Secure Code Training

Most security tools only help developers fix vulnerabilities after they're already introduced. This results in piecemeal learning that occurs through trial and error, if at all. Helping developers learn to recognize common vulnerability patterns, understand attack vectors, and implement proper defensive measures can provide a greater impact with more comprehensive coverage of essential security concepts.

In addition, structured training can ensure consistent security knowledge across development teams. When all developers share a common understanding of security principles and best practices, they can better collaborate on security issues, conduct more effective code reviews, and establish shared secure coding standards. This collaborative environment helps create a security-minded culture where secure coding becomes a natural part of the development process rather than an afterthought.

06

Cultivate Security Champions

Security champions serve as vital bridges between security and development teams. As developers, they have a deep understanding of their team's codebase and challenges and can provide immediate guidance and support to other developers. By speaking the same technical language and sharing similar daily experiences, they're often more effective at conveying security concepts than external security teams.

Security champions are an impact multiplier for security awareness and adoption among developers. As respected members of their development teams, they influence security culture from within, making secure coding practices feel less like external mandates and more like natural engineering best practices. Their success stories and practical solutions can be shared across teams, creating organic growth in security adoption throughout the organization.

07

Define Joint Metrics to Track Progress

What gets measured gets managed: Metrics shape behavior by focusing attention on what matters most, so choosing the right metrics is essential. Key security metrics often include Mean Time to Remediate (MTTR), which tracks the average time to fix vulnerabilities, and Change Failure Rate, which measures the percentage of releases requiring security fixes post-deployment.

However, security teams should avoid focusing solely on 'negative' metrics like vulnerability counts. 'Positive' metrics, like secure development training completion, developer adoption of security tools, and proactive security design reviews, can encourage teams to build security in from the start rather than treating it as an afterthought. And remember – metrics shouldn't just be for developers. Metrics like Lead Time can encourage security teams to ensure that their security tools and practices meet the needs of development velocity.

Going From Found to Fixed

The path to unlocking developer adoption is not about forcing compliance, but rather empowering developers to view security issues as part of their scope of work. Consequently, they will fix security issues faster and more effectively. By making security accessible, actionable, and aligned with development practices, organizations can create a sustainable security program that keeps pace with evolving threats without slowing down development.

Introduce AppSec that empowers developers with

The AppSec Platform for DevSecOps

Request a Demo

Checkmarx

Checkmarx helps the world's largest enterprises get ahead of application risk without slowing down development. We end the guesswork by identifying the most critical issues to fix and give AppSec the tools they need, all while letting developers work the way they want. From DevSecOps to developer experience, security and development teams can now work better together. That's why 1700+ customers rely on Checkmarx to scan over 1 trillion lines of code annually, improve developer productivity by 50%, and deliver 2X AppSec ROI.

Checkmarx. Always Ready To Run.