An Integrated Approach to Embedding Security into DevOps
A Best Practices Guide

Software = Security
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>A Brief DevOps Primer</td>
</tr>
<tr>
<td>Different AST Solutions Find Different Software Vulnerabilities</td>
</tr>
<tr>
<td>Where to Embed AST Solutions into DevOps</td>
</tr>
<tr>
<td>An AppSec Awareness Program Makes the Most Sense</td>
</tr>
<tr>
<td>An SDLC Integration and Orchestration Layer is Critical</td>
</tr>
<tr>
<td>Automation of AST Solutions within DevOps Tooling is of Upmost Importance</td>
</tr>
<tr>
<td>Managing and Reducing Security Risks at Scale</td>
</tr>
<tr>
<td>Define Security Policies</td>
</tr>
<tr>
<td>Automate and Integrate</td>
</tr>
<tr>
<td>Identify Vulnerabilities</td>
</tr>
<tr>
<td>Correlate Results</td>
</tr>
<tr>
<td>Remediate Vulnerabilities</td>
</tr>
<tr>
<td>Manage and Monitor</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
<tr>
<td>About Checkmarx</td>
</tr>
</tbody>
</table>
Introduction

Organizations are adopting DevOps as a development and operational model to facilitate the practice of automating software delivery and deployment. With this shift, security and development leaders are finding that their traditional approaches to software security are not able to adapt to this new model and security is often viewed as an inhibitor to DevOps. This does not need to be the case. With the right tool, services, and processes in place, security can fit right into a DevOps environment. In this Guide, we'll delve into the absolute best approach to embedding automated security scanning into DevOps.
A Brief DevOps Primer

The DevOps movement established a culture and atmosphere whereby developing, testing, and delivering software was intended to take place quickly, regularly, and with more dependability. This cultural shift drove the inception of continuous integration (CI) and continuous delivery (CD) fundamentals, which are part of the DevOps building blocks today, as shown in Figure 1 below.

Figure 1

The DevOps Building Blocks

Fundamentally speaking, DevOps is about processes, connections, automation, and tooling throughout the development, test, and delivery stages. But more importantly, DevOps is about the “automation of tooling” and the different “tooling” associated with building software. However, one thing that DevOps fundamentals have failed to address on their own is, where to embed software security throughout the entire software development ecosystem. For organizations that desire to produce more-secure software, the use of multiple Application Security Testing (AST) solutions is imperative within DevOps to address the vulnerabilities found in uncompiled code, running code, and open source components. Let’s delve into why that is, while exploring the various AST solutions on the market today.
Different AST Solutions Find Different Software Vulnerabilities

Static Application Security Testing (SAST) solutions are used to incrementally scan (test) uncompiled code for vulnerabilities during the software development process itself within Dev. The code is still in its uncompiled state and static testing is designed to find flaws like SQL injection much more easily. SAST solutions are great at providing code-level guidance as to where and how to fix vulnerabilities in source code. SAST fits well into integrated development environments (IDEs), issue trackers, and build tools to support CI/CD workflows. SAST fits well in DevOps since it doesn’t introduce significant delays.

Interactive Application Security Testing (IAST) solutions are better at detecting deployment configuration flaws in running applications found during functional testing—before the application is deployed. It would be imprudent to assume that applications will be vulnerability-free after the development phase, or that code in run-time doesn't need to be tested. IAST understands how all the pieces of an application work together and operate at run-time, so it can detect vulnerabilities in running applications that attackers may be able to exploit.

IAST fits well into DevOps since it doesn’t introduce delays beyond the time needed to perform functional testing.

Software Composition Analysis (SCA) solutions empower development, security, and operations teams with the insight necessary to efficiently address the risks associated with the open source software within the applications they create, deploy, and maintain. Analyzing and managing open source components in use ensures that vulnerable components are removed or replaced before they become a problem. SCA fits well into DevOps since it doesn’t introduce significant delays.

Dynamic Application Security Testing (DAST) tools detect vulnerabilities on running applications by externally attacking the application. DAST coverage is limited to reflective types of vulnerabilities, since DAST solutions are essentially blind as to what is happening inside an application. DAST test results offer no code-level guidance as to where software vulnerabilities are located, making it difficult for developers to easily fix identified vulnerabilities. DAST tools can't effectively achieve the fast turnaround times the DevOps requires.
Pentesting Approaches must also be addressed herein, although this type of testing generally falls outside of AST solutions in general. Many organizations take this approach as a means of ensuring their applications are free of vulnerabilities, but it happens very late in the SDLC, even after using DAST solutions. This approach is often the result of the need to “check a box”, so to speak, and the results of this testing have very little ability to provide guidance to developers as to what needs to be fixed.

Falling within the security-gate mentality, the time between pentests is often very lengthy, and this type of testing is not performed repetitively during software development, but is done after applications are often live, or ready to go live. Attacking the application from the outside-in does result in the detection of many true positives, but it does not help developers see where the problems are in the code they create.

Now that we’ve covered the various AST solutions, let’s explore where to embed AST solutions within DevOps next.
Where to Embed AST Solutions into DevOps

In Figure 2 below, the various AST solutions just discussed are shown within Dev on the left and encroaching into Ops on the right. This figure should help organizations understand where various AST solutions fit within the stages of DevOps.

As we can see in Figure 2, AST solutions must be embedded within the stages of Dev, while encroaching into Ops. The correct AST solutions for each stage of Dev are highlighted by the thin green lines around the various stages. The bullets below highlight the exact stage (or stages) where AST solutions fit best within Dev as well:

- SAST operates throughout the CODE, CHECK-IN, BUILD, and TEST/QA stages
- IAST operates during the TEST/QA stage
- SCA operates during the BUILD and TEST/QA stages
- DAST operates during the TEST/QA stage, but has limitations as previously mentioned.
- Pentesting primarily operates outside of software development altogether.
Although not covered yet, in Figure 2, AppSec Services (the outer green line) allows organizations to outsource some of their requirements to a third-party that helps introduce and implement application security processes, secure coding practices, security testing, and vulnerability remediation. This can also include kick-start programs designed to help implement and integrate security solutions into the SLDC, consulting on security findings, instruction on how to reduce false positives, training, threat modeling, and more. Organizations that lack internal resources and expertise during the initial phases of DevSecOps can benefit from these services.

AppSec Professional Services can be tailored and tuned to an organization’s unique technical and strategic business requirements. They’re also designed to enhance secure DevOps initiatives and strengthen application security postures, accelerate on-boarding and implementation to realize benefits earlier, and refine testing procedures and configurations.

Also, in Figure 2, a new term (SCE) has been added that has not been covered yet. SCE stands for Secure Coding Education (also known as secure code training) and operates within the CODE stages of DevOps. The next section briefly highlights SCE within the context of an AppSec Awareness Program in general.
To become a security-focused developer is not only about secure coding education or training. It’s more about operational awareness, where everybody in the organization understands the importance of security. As developers start to write code, they should have security entrenched in their minds. No longer should AppSec teams be the only group responsible for security. Instead, developers also need to responsible for security, and as a result, will be capable of writing more-secure code. When everything today is moving to code, (e.g., Infrastructure as Code,) everyone needs to have security as top of mind, including management, developers, AppSec teams, and even IT teams.

However, the reality is that a significant percentage of developers don’t have confidence in the security of their own applications, or they have little if any intimate knowledge of vulnerabilities and how they’re created. This gap exists because developers are measured by speed and the number of functional bugs in their code, not the amount of security vulnerabilities they induce.

To bridge this gap, organizations now understand that they need to provide their developers SCE (in the context of an AppSec Awareness Program) that’s incorporated right into their IDEs. Using just-in-time training solutions, ongoing communication, and fun engagement, security managers cultivate a culture of software security that empowers developers to think and act securely in their day-to-day work. Developers that think and act security can measurably increase the security of their software, reduce repetitive coding errors, and significantly lessen the number of software vulnerabilities that must be triaged and fixed.

In comparison, the problem with traditional secure coding education and training methods such as video tutorials, periodic classroom training, and mandatory online courses often fail to achieve secure coding practices, since they are mundane, out of context, and not interactive. The whole idea of AppSec Awareness is to enhance the security maturity of developers and must become an organizational goal.

Since DevSecOps means more than just embedding AST solutions into DevOps processes, let’s review a few more significant areas of focus concerning the importance of automation in DevOps.
An SDLC Integration and Orchestration Layer is Critical

Adding an SDLC Integration and Orchestration layer to the AST solutions mentioned above helps unify the solutions into an integrated and easy-to-use platform that's designed to provide organizations with a holistic view of their software vulnerabilities at scale and simplify AST automation. As a result, this enables organizations to easily track, manage, and remediate security risks at scale. This layer is vastly needed to simplify and centrally managed end-to-end automation and orchestration flow, from scan to ticketing. When looking to implement AST solutions, ensure they come with an SDLC integration and orchestration layer as part of the solution.

Automation of AST Solutions within DevOps Tooling is of Upmost Importance

Obtaining DevSecOps requires organizations to automatically incorporate AST solutions throughout DevOps to eliminate manual testing procedures that may have caused delays in the past. These AST solutions must be as transparent as possible to developers and security teams ensuring the agility of DevOps is not hindered. Automation is key to helping fulfill regulatory requirements as well as managing overall risk. In order to meet this objective, AST solutions must be capable of being completely automated within the tooling that's often already in use within DevOps. Beyond automation and tooling, the next section expands upon the activities needed in order to manage and reduce security risks at scale. Let's explore that topic next.
Managing and Reducing Security Risks at Scale

When discussing where to embed security into DevOps to achieve DevSecOps, there are several aspects that are somewhat beyond application security testing in general, while others are directly related to it. Figure 3 highlights the activities that must be performed in order to fully manage and reduce security risks at scale. Starting at the top and moving clockwise, let’s look at each one of these concepts next.

Figure 3
Define Security Policies

This is where organizations define their application security (AppSec) policies concerning what are the acceptable and non-acceptable risks they’re willing to take. Applications will always have vulnerabilities and no organization will ever fully achieve zero vulnerabilities and zero functional bugs. Defining what risks are acceptable, and what are not, is imperative at this stage. And the whole point is to determine what policy violation(s) would cause an organization to break a build.

The security policy that is defined also serves as a pseudo contract between AppSec teams and developers, so both fully understand what’s expected of them in terms of security. This policy also serves as guidance as to what vulnerabilities should be remediated first as a result of application security testing. Defining security policies is tightly associated with DevSecOps and these policies are vital to measuring the overall success of your DevSecOps initiatives.

Automate and Integrate

This is where organizations perform the activity of integrating their SAST, IAST, and SCA testing solutions into their build/development environments—making sure that the AST scans are completely automated. Without automation, organizations cannot scale. Each organization can choose to what level they desire to automate, since it can be done in many ways and forms. But eventually you want to make sure that your applications are being scanned in a consistent manner. The best way to do that is to automate the scans within the build environment, the development environment, or both.

For example, you want to make sure that you automate your AST solutions when the builds are running in the build environment or when developers are performing a code commit or a pull request, etc. In the latter case, that automation takes place earlier in the development environment.

When AST solutions are being automated into the coding phase, development teams use self-service to automate scans via code collaboration platforms such as GitHub, Azure DevOps, etc. When AST solutions are being automated during the build/CI phase, CI plugins are being used to automate the scans. Finally, ticketing system integration closes the loop by handing developers the relevant findings from their scans in real-time.
Once you've performed the activity of integrating and automating the AST solutions as previously described, this step is where the AST scans are being performed. Using SAST, IAST, and SCA in an automated fashion, these solutions are fully capable of detecting all sorts of vulnerabilities in your software applications. These can include vulnerabilities in:

- Uncompiled Code
- Running Code
- Open Source Components

The whole point is to detect coding errors (that may cause vulnerabilities) as early on as possible without slowing down the development, delivery, and deployment of software applications, ensuring the agility of DevOps is effectively maintained.

The idea behind correlation is to increase the level of confidence and priority of the high-risk findings (vulnerabilities detected) from AST solutions, especially when you're able to correlate the same findings from different scanning solutions. For example, if there was a SQL injection vulnerability discovered by SAST during static testing, and IAST confirms the same finding during interactive testing, if you can correlate both of those findings together, you can increase the confidence level that the finding is a true positive.

In this case, the likelihood of a finding being reproducible is extremely high. When this is so, the vulnerability needs to be fixed sooner, rather than later. When organizations have hundreds of applications and their AST solutions are detecting thousands of potential vulnerabilities, the ability to scale starts here—when organizations can make sense of the large amounts of data from their scan findings.
Remediate Vulnerabilities

Remediation has two aspects. One is what should be fixed, and the other is how to fix it. When referring to what should be fixed, in the context of scale, no developer can handle thousands of vulnerability findings. You need to make sure that you can prioritize all those findings in a way that a developer can digest them.

When developers receive all their vulnerability findings, similar to the way they get all of the defects from their current defect management tool, helps speed up remediation time and AST adoption. In addition, if developers receive best practices on how to fix a certain vulnerability, with a best fix location, enables them to quickly and efficiently fix the greatest number of vulnerabilities. Simply put, developers need to be able to focus on what’s most important, and to work on fixing the vulnerabilities that would exponentially reduce the most risk first.

Once a team decides what needs to be remediated, often based on the policy set forth in the first bullet (Define Security Policies), the next decision is how to remediate. Here is where Secure Coding Education (SCE) can be of great assistance. SCE can teach developers how to fix a certain vulnerability with a customized lesson that is specific to that type of vulnerability, especially if SCE is integrated directly into developers' IDEs.

Manage and Monitor

Manage and monitor are where organizations track their application security program's Key Performance Indicators (KPIs). This allows organizations to see if, over time, the amount of the vulnerabilities is decreasing, the rate of introducing new vulnerabilities is decreasing, and the rate of severe vulnerabilities is decreasing as well. There are all kinds of KPIs that organizations use to see if their security program is effective.

Part of that KPI cycle is about knowing what areas need improvement and what areas don’t. This allows teams to determine if the security policy is being met or not, or if developers need more training, tools, or incentives. Teams can also determine if the policy in place needs refinement, etc. All this activity allows organizations to measure their program's current status and level of improvements being made. It also creates a feedback loop, back to your developers since this process never ends. And the whole idea is to allow continuous improvement throughout your DevSecOps ecosystem.
Conclusion

The goals of this Guide are to foster a certain level of understanding pertaining to where to embed security into an organization's DevOps culture in the hope of helping organizations fully obtain DevSecOps. What's really achieved by embedding Sec into DevOps, in the most automated fashion as possible, is more-secure software that supports an organization's bottom line, while reducing risk.
About Checkmarx

Software security for DevOps and beyond.

Checkmarx makes software security essential infrastructure: unified with DevOps, and seamlessly embedded into your entire CI/CD pipeline, from uncompiled code to runtime testing. Our holistic platform sets the new standard for instilling security into modern development.

With Checkmarx you get:

Security from the Start
We deliver the industry’s most comprehensive, unified software security platform that tightly integrates SAST, SCA, IAST and AppSec Awareness to embed software security throughout the CI/CD pipeline and reduce software risk.

Simplified AST Automation
Checkmarx tightly integrates with common software release orchestration and agile planning tools, such as IDEs, build management servers, bug tracking tools, and source repositories to automatically enforce security policies.

DevOps Speed
Only Checkmarx enables you to manage software exposure at the speed of DevOps – getting applications to production quickly and securely without interrupting developer workflows.

Unmatched DevSecOps Expertise
We know software like no one else. We know security like no one else. Developers like Checkmarx better than anyone else.

To learn more about our Software Security Platform please go to www.checkmarx.com/products/software-security-platform

Software = Security