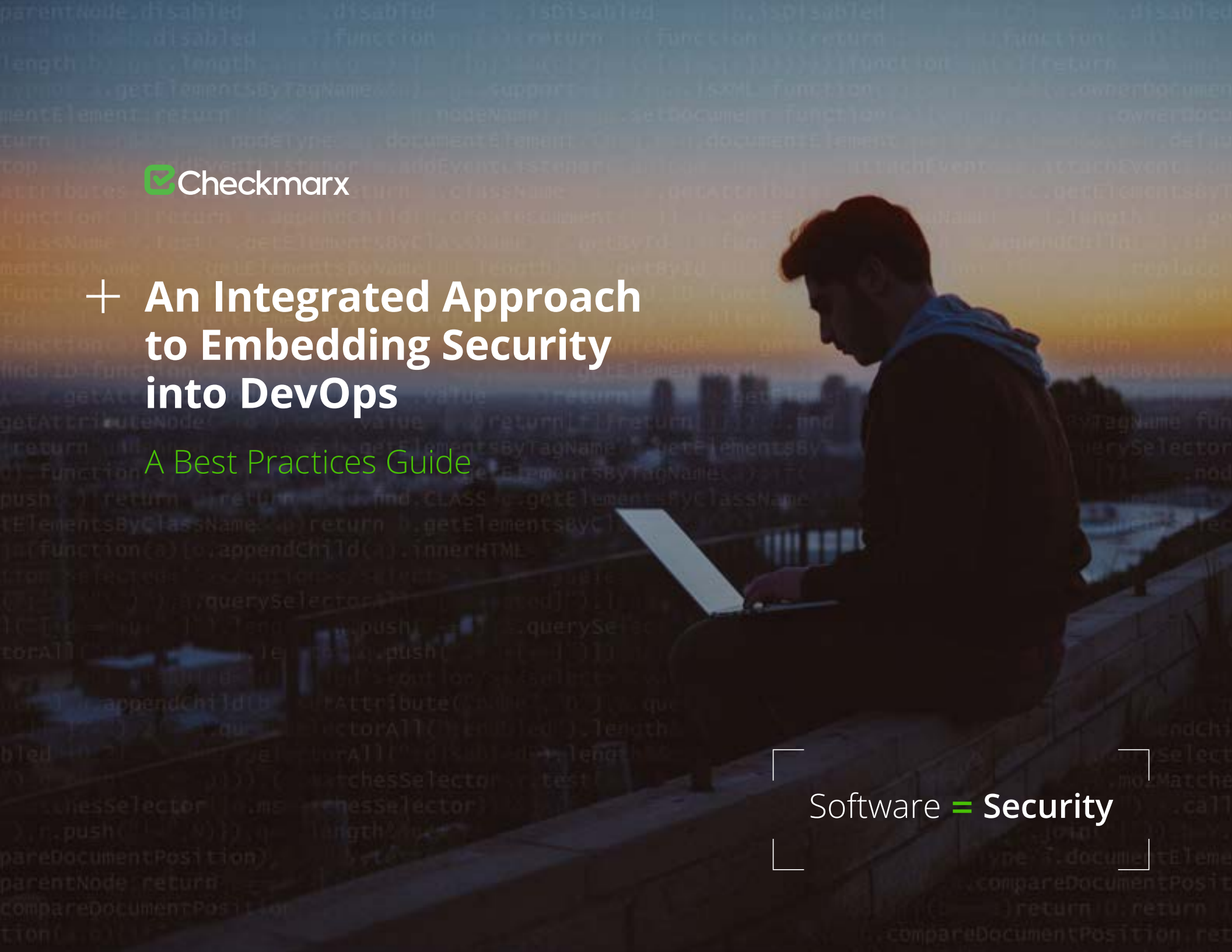


Checkmarx

An Integrated Approach to Embedding Security into DevOps

A Best Practices Guide

Software = **Security**

Introduction

What's making your software essential to your business, is also making it more dangerous. When software is everywhere, everything becomes an attack surface. The way your organization develops and depends on software has changed - and never has it exposed you to more risk. And while software security has never been more business critical, organizations know if it gets in the way of DevOps, it just won't work. Security must be inseparable from software development.

As the backbone of the digital transformation, software is becoming increasingly complex through interconnectivity, mobile, cloud, open source, IoT, microservices, and AI.
For example:

+ Over 80 percent of the code in today's software applications is open source
+ 30 billion connected IoT devices by 2020
+ 85 percent of customer interactions will be managed without a human by 2020

Despite this complexity, time to market is the new name of the game. For example, Facebook on Android alone does between 50,000 to 60,000 builds a day. However, this speed comes at a price. Verizon reports that nearly 70 percent of the data breaches they've investigated are due to attackers targeting vulnerable web applications, and risks are amplified as organizations move to DevOps without implementing proper security practices. In fact, 70 percent of today's developers indicate they lack the necessary training to adequately secure software.

The recent industry shift towards DevOps makes it clear that organizations are adopting this development and operational model to facilitate the practice of automating software delivery and deployment. As a result, organizations are acknowledging that their traditional approaches to software security are having a difficult time adapting to this new model, since security is often viewed as an inhibitor to DevOps. In this eBook, we'll delve deeply into the absolute best approach to embedding security into DevOps.

Contents

Why This eBook

The root cause of many successful cyberattacks lies primarily in vulnerable software itself. The real question that needs to be asked is, “Can the industry do a better job of writing more-secure code, making software applications nearly impenetrable to cyberattacks?” Here at Checkmarx we believe the answer is yes. Checkmarx is dedicated to building software security solutions that address the root cause of nearly every successful cyberattack by finding, classifying, reporting, and demonstrating where and how to fix vulnerabilities in software.

Understanding that organizations all over the world are on a steady path of adopting DevOps fundamentals within their software development and operational processes, the need for increasing levels of software security is becoming even more apparent. This eBook will clearly demonstrate how organizations can embed security throughout their DevOps initiatives and gain assurance that the software applications being released to the internet are secured against the broadening attack landscape found there.

Here at Checkmarx we recognize there is an essential gap pertaining to the industry’s understanding of where and how to embed security into DevOps, transforming it into the grand achievement of obtaining true **DevSecOps**. From our discovery and findings, we realize that organizations desire knowledge about how to migrate to DevSecOps, hence the reason for this eBook.

“

This eBook will clearly demonstrate how organizations can embed security throughout their DevOps initiatives and gain assurance that the software applications being released to the internet are secured against the broadening attack landscape found there.

What You Will Learn

In this eBook, we'll discuss some of the common software development methodologies of the past in the context of security overall. We'll call out what hasn't worked, what's not working so well today, and what organizations need to do to fully gain DevSecOps. We'll discuss where security enters the picture in DevOps and refute one of the common misconceptions being discussed in the application security industry. We'll cover many aspects of embedding security into DevOps, but more importantly, this eBook provides detailed information concerning *what to do, how to do it*, and even *why to do it.*

We'll also dive deeply into the importance of ***integrating and embedding*** Application Security Testing solutions into your DevOps initiatives. Finally, we'll thoroughly cover the critical importance of ***automation*** when trying to embed security into DevOps, since it's the most important aspect of obtaining DevSecOps. Automation is a key element that must not be overlooked. Understanding these concepts at length will allow organizations to make better decisions concerning their DevSecOps initiatives, and as a result, drastically improve the security of the software they produce.

By reading this eBook, you will gain a solid understanding of what steps your organization should consider in order to begin an integrated approach to embedding security into DevOps.

Chapter 1

Security within the Common Development Methodologies

The evolution of software development has taken many turns since its inception. FORTRAN, released by IBM in 1954, was the first commercially available and widely adopted programming language. Fast forward sixty-five years and today, programming languages abound. Although languages have changed dramatically, the primary purpose of software development hasn't.

In comparison to 1954, the approaches to the way software is developed have majorly changed. Software development methodologies have profoundly evolved for the sole purpose of improving the quality, accuracy, and speed of delivering and deploying software. Driving these changes are a combination of time to market, customer requirements, and business demands. With the dramatic rise in cyberattacks and the resulting data breaches making the news daily, security has now come to the forefront within the newer development methodologies.

From a software security perspective, let's observe the software development methodologies used in the past, where security came into play, where it didn't fit in so well, and where the industry is moving towards.

Security and Waterfall

For decades, the stages of software development were organized and accomplished using what was called a Waterfall methodology. The overall task was organized into stages that closely resembled a manufacturing assembly line, due to its proven success in producing goods of consistent quality, for the lowest cost. The assembly line was very linear in nature and anything that slowed or stopped the ***line*** equated to losses in production and revenues. However, where does adding security within the Waterfall methodology come into play?

Traditionally, Application Security (AppSec) teams worked with software project teams during two phases: technical requirements and design, and right before the software was scheduled to go live. After a project team gathered the business requirements and defined a target architecture, their project next went to an AppSec team for review. This review usually included some form of threat modeling and data flow exercises. Once complete, a list of security requirements was noted, and they were placed into the project design document. The document was handed over to development teams, who in turn worked on the application development for the next several months (or longer).

After development teams had a working application that met the documented requirements, and users agreed it's what they needed, the project team would engage with AppSec teams once again. Operating as a ***security gate***, AppSec teams would run various penetration tests to find any security vulnerabilities in the draft-final product. They would likely

catalog a long list of vulnerabilities that had to be fixed before allowing the software to go live, and effectively halted the assembly-line process. See Figure 1 below that highlights where the location of the ***security gate*** was in the Waterfall methodology.

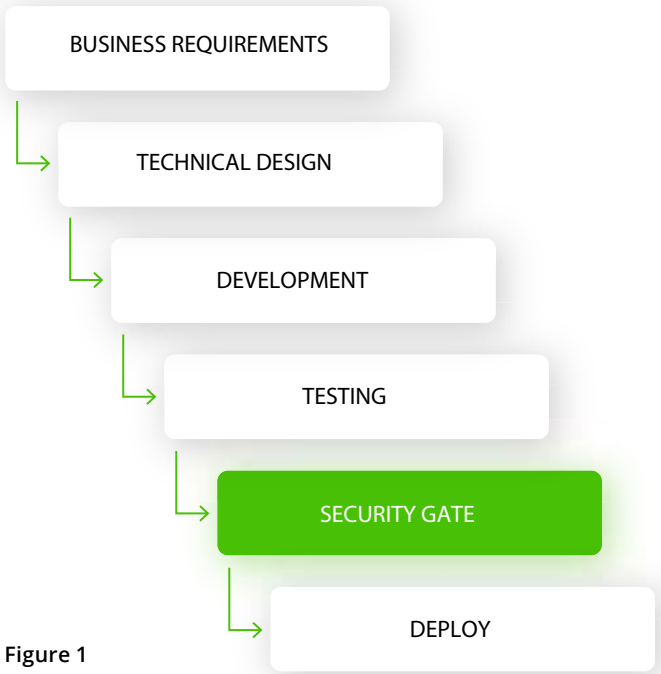


Figure 1

Unfortunately for everyone involved, this approach engaged AppSec teams near the end of the assembly line. The delay caused by this approach, intending to give developers time to fix security issues near the end of the project, certainly jeopardized the project schedule overall.

As a result, everyone was unhappy. AppSec teams were unhappy because identified security issues would likely make their way into production, since the project ***schedule*** would usually win over ***security***. Development teams were unhappy because they had to work overtime fixing many of the discovered security issues. Finally, leadership was unhappy because it appeared that project teams, AppSec teams, and development teams did not work well together. There is, of course, a better way.

Security and Agile

It was clear that a better methodology was needed that could incorporate security into the development processes, but was Agile created to address security? Not necessary so.

Agile came about to address the linearly-based shortcomings of the Waterfall methodology. In 2001, a group of influencers began to rethink software development and as a result, produced the Agile Manifesto.

The whole point of Agile was to create an agile (responsive to change) environment that did not replicate a completely linear, non-flexible, assembly-line process like Waterfall. The most important characteristic of Agile methodologies allowed developers to become more flexible to change during the overall process, based

on iterative (repeating, repetitive) needs and requirements of the business. Collaboration between cross-functional teams was highly emphasized in Agile methodologies, since it empowered people to make team decisions and adjust to continuous planning, testing, and integration requirements.

The Software Security Sandwich and Where It Comes From

In the both the Waterfall and Agile methodologies there are many security-related activities during the business requirements and technical design stage. There are also many security-related software testing processes performed before the application is scheduled to go live. These two activities, often occurring months apart, results in a ***software security sandwich*** so to speak. Although Agile has been widely accepted and is being used by development teams all over the world today, it still does not effectively solve the software security sandwich that existed in Waterfall. This sandwich still exists because organizations have not integrated security into all stages of software development. Agile certainly helps address the transitional nature of today's business cycles, but it still doesn't always address security head on.

Agile certainly helps address the transitional nature of today's business cycles, but it still doesn't always address security head on.

Security and DevOps

As agile approaches became more commonplace, a philosophical shift began to be observed. Once development **(Dev)** teams completed their work, their project was turned over to an operations **(Ops)** team for deployment, support, and ongoing maintenance. However, what was experienced during that early era was that software would often run fine in the developers' environments, but it had issues running in production. This created lots of friction between Dev teams and Ops teams, since neither team had intimate knowledge of what the other team did or was tasked with doing.

Near the end of the 2000's, the industry saw the birth of **DevOps** due to a group of stakeholders who began to confer about how to build better linkage between Dev teams and Ops teams; creating new practices that drove a shift in traditional thinking. 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 2.

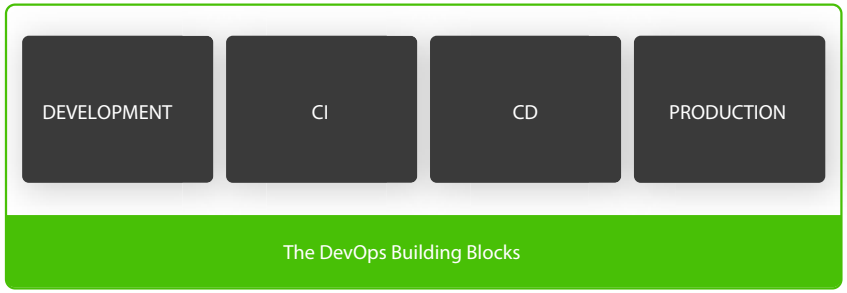


Figure 2

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, *how to embed software security throughout the entire software development ecosystem.* Concerning software security in both Agile and DevOps, there's been lots of industry chatter over the last few years about a new method of adding security into these environments called "shift left". Let's look at this concept next.

Shift Left vs. Shift Center

Within the software development industry, the term “shift left” surfaced as a result of organizations waiting to perform security testing until the end of the development process, often causing unexpected delays. If you’re testing your software looking for security vulnerabilities near the end of the development process (on the right), the recommendation is to ***shift your testing further to the left*** and perform security testing sooner—hopefully reducing delays. However, this makes very little sense overall, since DevOps is not linear like the Waterfall methodology. DevOps is circular as depicted quite well in Figure 3 below.

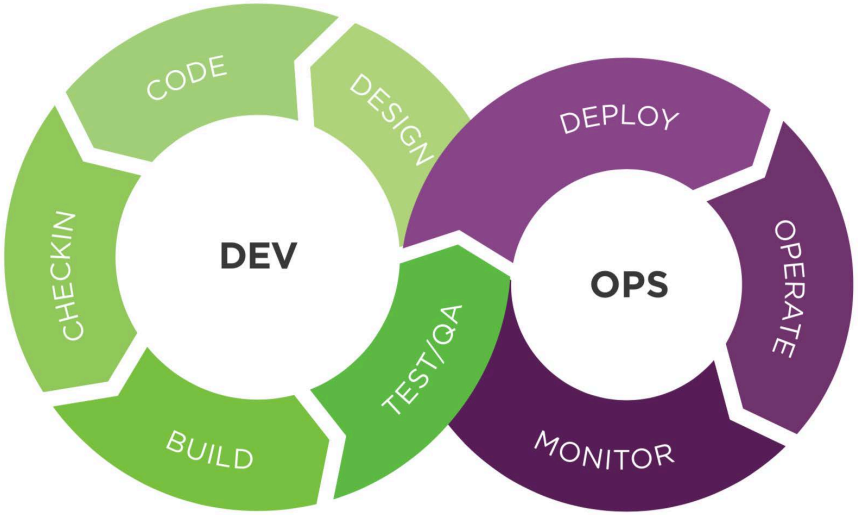


Figure 3

As we see in Figure 3, DevOps really doesn’t have a left or right in comparison to the more-linear processes used in Waterfall (Figure 1). Sure, **Dev** is on the left and **Ops** is on the right, but DevOps is more like a figure-8 infinity loop that has no beginning and no end. The Dev process never stops, and the Ops process never stops as well. If someone were to shift left in Figure 3, where is left?

A better recommendation would be to ***shift center*** and embed software security solutions throughout DevOps. We’ll discuss shift center more in depth in Chapter 2. However, here is an analogy that may help make better sense of the concept of "shift left" and why it really doesn’t fit in DevOps.

Linear vs. Circular – An Analogy That Should Help

When you're on a train and someone wants to get on or off the train, what does the train do? The train stops, passengers get on, and passengers get off. Then the train starts to move, only to do the same thing farther down the track. This is not the way we want to add software security to our DevOps initiatives, since it would most likely be a disaster. The whole point of CI and CD is that the *train* is never supposed to stop and shifting left just doesn't fit. Instead let's look at a better analogy that may hint of a new method of adding security into DevOps processes.

The London Eye is one of the world's tallest Ferris wheels. The interesting part about the London Eye is that when it's open and running, the Ferris wheel never stops to on-board or off-board passengers. People get on, people get off, but the wheel continues without hesitation. No one on the ride knows who got on or who got off the other "pods", since from their perspective, there's no impact to their own ride experience. This is the way organizations need to think about adding software security into DevOps environments—in a way that never stops the process.

We are somewhat refuting the concept of "shift left", since it doesn't make sense when observing the double-helix, infinity loop represented by Figure 3. There really is no *left* in DevOps. Therefore, how should organizations embed security within DevOps? The next section will highlight what's needed.

> This is the way organizations need to think about adding software security into DevOps environments—in a way that never stops the process.

+ Chapter 2

Embedding Security into DevOps

Statistics demonstrate that a high number of security breaches resulting in the theft of private data are derived from attackers taking advantage of vulnerable software. Since that is the case, what's needed to solve the problem of vulnerable software? Simple. Organizations must find a balance between speed and security within DevOps. Let's look at the current approach to security within DevOps.

The Current Approach to Security within DevOps

When discussing security within DevOps, let's first define what *security* is. In the world of software development, test, and operations, security can mean many different things. From defining security policies, automating security testing, identifying vulnerabilities, correlating results, and remediating vulnerabilities, to management and monitoring of security programs, and developer' KPIs, there are many stages of security in an organization. However, let's look at ***functional testing*** and ***application security testing*** first.

Functional testing is where software is tested against a list of functional criteria to ensure the software operates as intended. This testing is normally performed through various automated and / or manual procedures, but the whole idea is that software must be operational before it is deployed. Functional testing is performed right before the software application is scheduled to go live.

“

From defining security policies, automating security testing, identifying vulnerabilities, correlating results, and remediating vulnerabilities, to management and monitoring of security programs, and developer' KPIs, there are many stages of security in an organization.

The second type of testing is known as ***Application Security Testing (AST)***, and this is where identifying vulnerabilities, correlating results, and remediating vulnerabilities must be performed before the software is deployed. Today, and in lots of DevOps environments, both types of testing (functional and AST) are performed within the Test/QA stage shown in Figure 3. And this is where the ***need for speed*** and the ***need for security*** tend to clash. The reason for this is that AST has not been embedded into the entire Dev process. Instead, and in most cases, AST only exists in one place—at the transition point where software moves from Dev to Ops, often creating a bottleneck that induces delays.

Development teams are driven by time to market. AppSec teams are driven by functioning software that's secure. Here is where a balancing act is taking place. AST cannot delay deployment, and if it does, it usually means time wins over security. Added delays for AST, vulnerability triage, and remediation somewhat defeats the whole precept of CI/CD. There must be a better way of performing AST throughout the entire DevOps process that will not invoke delays. In the next section, we'll discuss a better approach.

A Better Approach to Security within DevOps

When reviewing Figure 3, there once again appears to be part of a software security sandwich called ***Test/QA***, operating somewhat similarly to the Waterfall methodology and its security gate mentality.

“

Development teams are driven by time to market. AppSec teams are driven by functioning software that's secure. Here is where a balancing act is taking place.

> Development teams are driven by time to market. AppSec teams are driven by functioning software that's secure. Here is where a balancing act is taking place. AST cannot delay deployment, and if it does, it usually means "time" wins over "security".

However, there is a way of embedding AST solutions throughout all the Dev stages, but how is that done? In Figure 4 below, the concept of shift center may make better sense.

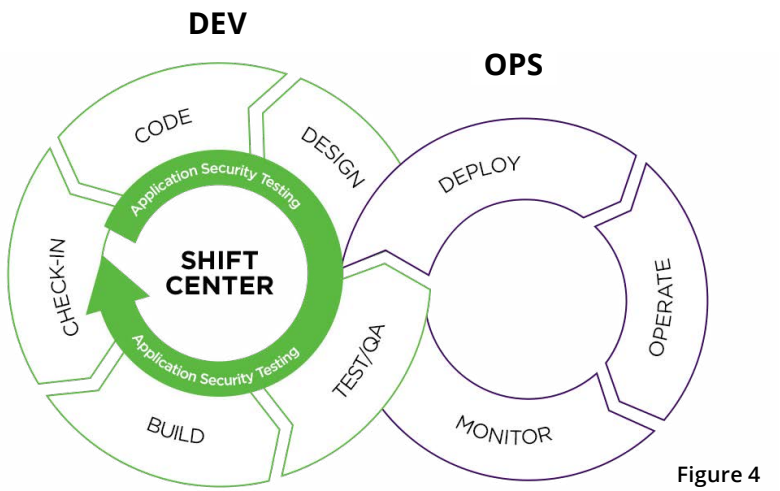


Figure 4

As shown in Figure 4, AST solutions can be embedded into the stages of Dev to include Design, Code, Check-in, Build, and Test/QA. Let's quickly list some of the available AST solutions on the market today.

AST Solutions:

SAST – Static Application Security Testing
IAST – Integrated Application Security Testing
SCA – Software Composition Analysis
DAST – Dynamic Application Security Testing

In Chapter 3, a more in-depth discussion of the available AST solutions listed here is provided, in addition to where they fit within DevOps.

These Topics Must be Addressed when Embedding Security into DevOps

Beyond Application Security Testing (AST), there are many things that must happen within an organization that is embracing DevOps initiatives. Remember, DevOps is a cultural shift that goes beyond just adding AST to your DevOps environments. With this in mind, let's discuss several other topics that must be addressed before we move to the next chapter.

Paying Attention to Open Source Vulnerabilities in DevOps

The adoption of open source components by software development teams dramatically changed the software industry. Instead of building all software *from scratch*, organizations use open source components to provide common or repetitive features and functionalities. This limits the use of custom code to proprietary features and functionality. Open source software is still software and it's exposed to coding errors that can result in security vulnerabilities. Addressing vulnerabilities in open source components is critical and must become part of the DevOps process.

Open source software is still software and it's exposed to coding errors that can result in security vulnerabilities.

Recognizing the Issue of Code Complexity in DevOps

Another significant contributing factor to developers introducing vulnerabilities is due to ***code complexity***. Organizations with very large software applications typically do not have one person on staff that understands the entire code base, which can contribute to the propagation of security issues throughout a code base.

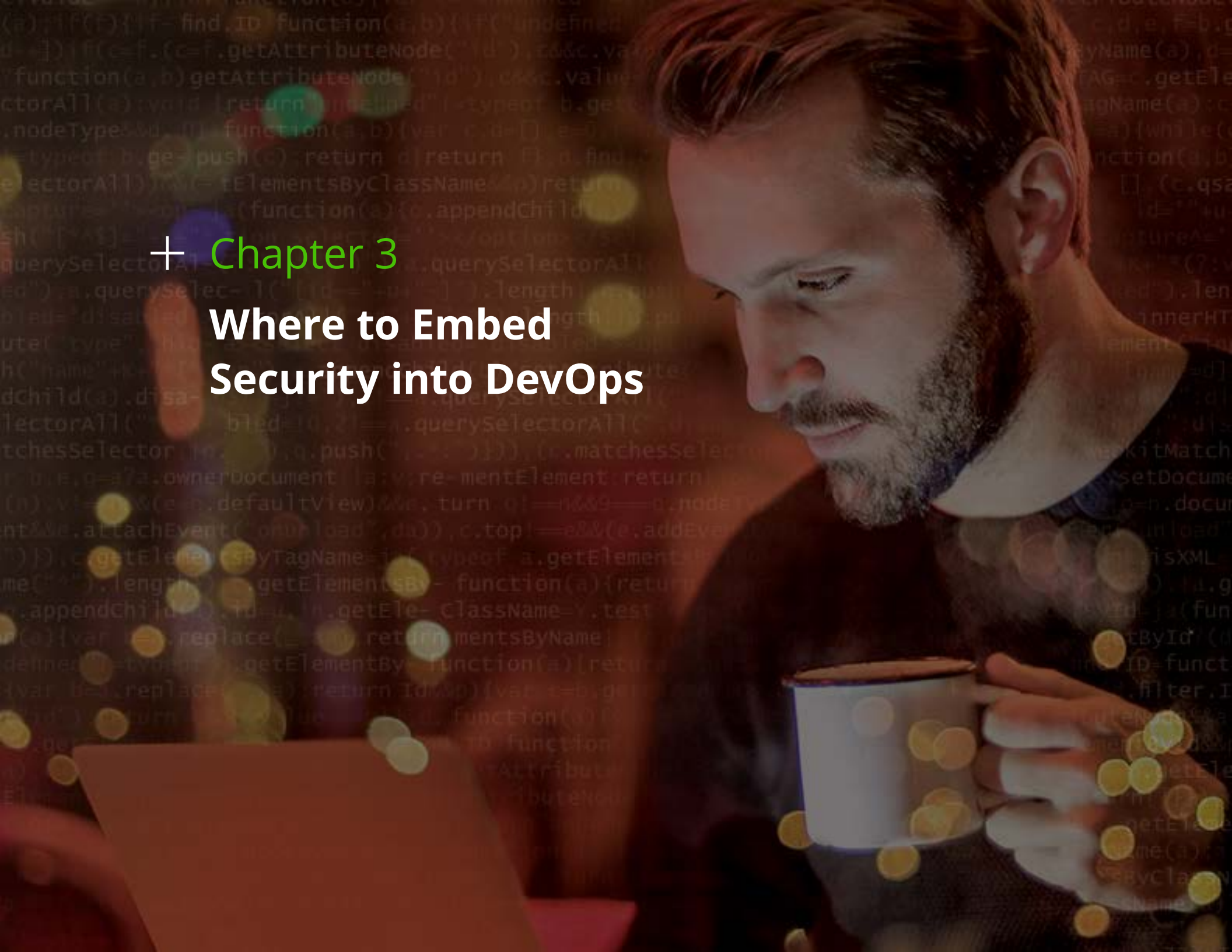
Today, it's very rare that home-grown software applications are built from scratch. They are normally a branch or a copy of an existing code base, and seldom do developers have ***all*** the code in their native work environments for the very large and complex software they work on. This concept of code complexity must also be addressed within DevOps, since complexity can contribute to repetitive vulnerabilities making their way into production.

Addressing Software Exposure While Not Impacting Speed

Software exposure results from mistakes made in the design, coding, testing and maintenance of software. Exploiting these vulnerabilities can make the software unavailable or unreliable to users, or allow attackers to execute unauthorized code, read or modify data, change a user's privileges, hide activities, or bypass security controls.

Since traditional application security and application testing approaches can only address specific facets of development at minimum speeds, they often miss vulnerabilities before they're released—slowing time to market and creating costly inefficiencies. While software security has never been more business critical, if it gets in the way of DevOps, it just won't work. The need for fast, incremental, static, run-time, and open source testing is critical to address software exposure at the speed of DevOps.

Now that we've discussed what's needed within the context of software security and DevOps, in the next chapter we'll discuss *where* and *how* to embed security into DevOps. We'll also discuss the many building blocks needed to achieve DevSecOps and why they're important.



Chapter 3

Where to Embed Security into DevOps

For nearly two decades, cyber attackers have willfully profited from organizations releasing vulnerable software on the internet. As a result, organizations have attempted to protect their vulnerable software with a host of different technologies that fall under the umbrella of "software security". From anti-malware solutions, DDoS defenses, next-gen firewalls, DMZs, and intrusion prevention systems, to web application firewalls (WAF), bot management solutions, runtime application self-protection (RASP), load-balancers, and a host of other technologies, none of these actually remedy the root cause of nearly every successful cyberattack—vulnerable software that is ripe for attack.

The products mentioned above attempt to manage the symptoms or the results of coming under a cyberattack, since none of them can find and fix the many exploitable vulnerabilities in software. ***Attempting to protect vulnerable software and applications from the outside-in is simply failing.*** There is a better way of securing software from the inside, by testing, finding, and remediating vulnerabilities, resulting in producing more-secure software. The next section outlines how that can be done today.

Addressing the Three Categories of Vulnerabilities Commonly Found in Software

Although the most common software weaknesses (or software errors) are enumerated in the OWASP Top 10 and the SANS Top 25, these entire lists can be broken down into three primary categories of vulnerabilities found in:

+ **Uncompiled Code**
+ **Running Code**
+ **Open Source Components**

For organizations who desire to produce more-secure software, the usage of multiple AST solutions is imperative within DevOps to address the vulnerabilities that likely exist in an organization's proprietary software. Let's delve into why that is.

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 required. DAST does not fit well into DevOps since it often introduces lengthy delays.

An Integrated Management and Orchestration Layer is Critical

Adding a ***Management 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. As a result, this enables organizations to easily track, manage, and remediate security risks at scale. This layer is vastly needed to unify security policies, gain full visibility of all vulnerabilities, optimize remediation efforts, and centralize user / role management of the AST solutions deployed. When looking to implement AST solutions, ensure they come with an integrated management and orchestration layer as part of the solution.

> This layer is vastly needed to unify security policies, gain full visibility of all vulnerabilities, optimize remediation efforts, and centralize user / role management of the AST solutions deployed.

Where to Embed AST Solutions into DevOps

On the following page is the same figure as shown in Figure 4. However, the addition of various AST solutions has been added to Figure 5 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 5, AST solutions must be imbedded 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

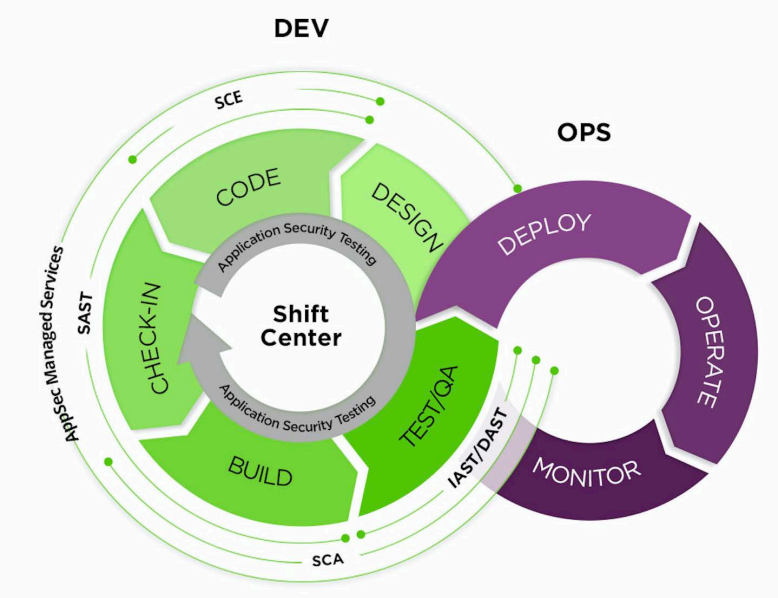


Figure 5

In figure 5, AppSec Managed Services allows organizations to outsource their AST to a third-party that helps introduce and implement application security processes, secure coding practices, security testing, and vulnerability remediation. Organizations that lack internal resources and expertise during the initial phases of embedding security into their DevOps environments can benefit from these services.

Also, in Figure 5, a new term (SCE) has been added that has not been covered yet. SCE stands for Secure Coding Education and operates within the CODE stages of DevOps. The next section will highlight SCE more in depth.

Shift Left in Education with SCE Makes the Most Sense

We've somewhat refuted the concept of shift left concerning AST solutions in DevOps. However, shift left does make sense when speaking to Secure Coding Education (SCE). The earlier you discover software vulnerabilities, the easier and less expensive they are to fix. 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 that's incorporated right into their IDEs as early on as possible. The problem with traditional 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 SCE is to enhance the ***security maturity*** of developers, enabling them to take an active role in developing more-secure code. Developers that ***think 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. When a vulnerability is found during AST, developers should have a way of highlighting the vulnerability, jump to a training lesson specifically about that type of vulnerability, and learn how to immediately fix the vulnerability, all while never leaving their IDEs.

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.

The whole idea of SCE is to enhance the security maturity of developers, enabling them to take an active role in developing more-secure code.

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 importance of automation in DevSecOps overall.

Managing and Reducing Security Risks at Scale - Automation is Key

As previously mentioned, there are many facets of software security in organizations today. When discussing embedding 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 6 highlights the activities that must be performed in order to fully manage and reduce security risks at scale.

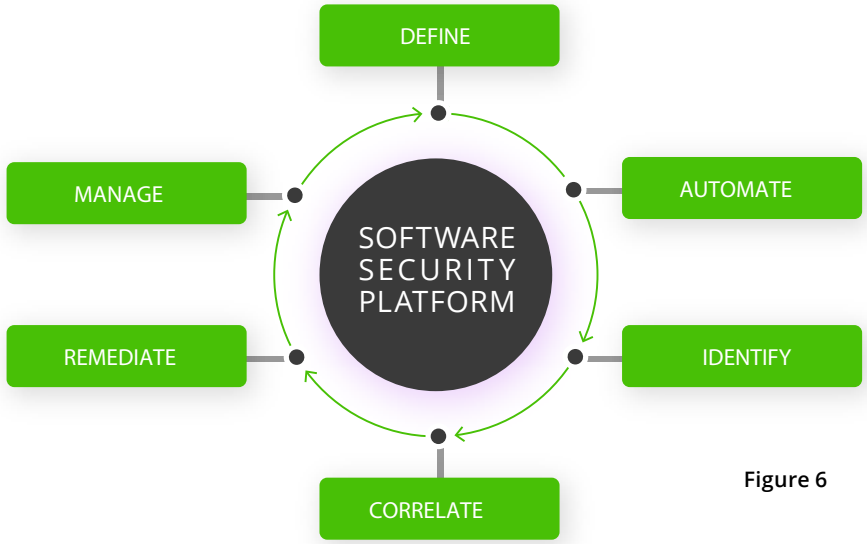


Figure 6

As shown in Figure 6, when each of these activities are being performed in the most automated and integrated fashion as possible, organizations begin moving towards a complete *Software Security Platform* that embeds security throughout DevOps. Next, let's explore each of the activities shown in Figure 6 in more detail.

+ Define Security Policies

This is where organizations define their application security policy 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.

The security policy that is defined 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 and / or 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

> This is where organizations perform the activity of integrating their SAST, IAST, and SCA testing solutions into their build and / or development environments—making sure that the AST scans are completely automated.

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.

+ Identify Vulnerabilities

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

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.

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.

+ Correlate Results

The idea behind ***correlation*** is to increase the level of confidence and priority of the high-risk findings 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

“

The idea behind correlation is to increase the level of confidence and priority of the high-risk findings from AST solutions, especially when you're able to correlate the same findings from different scanning solutions.

findings in a way that a developer can ***digest*** them. 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.

Having the ability to set an ***automated prioritization mechanism*** across the thousands of findings is of upmost importance. Using a set of criteria, organizations can define what's more important and what's less important. For example, a newly discovered open source vulnerability may be more important than a vulnerability in custom code. Fixing a vulnerability that has been in place for more than 60 days may be more important than something newly discovered. There are lots of criteria that will enable you to control how the findings are eventually ***prioritized***. When that is done in an automated manner, you can scale, and each developer receives their segment of code that they need to fix in an automated fashion. Today, few organizations are doing manual correlation. Either they have tools to correlate, or simply put, they are not correlating.

Typically from the ***automated prioritization mechanism***, it could also include machine learning algorithms that would be designed to focus your attention on what are the ***true positives***. Machine learning algorithms have the ability of setting ***percentage weights*** to various findings. For example, machine learning algorithms can be taught to understand that one type of vulnerability has an extremely high percentage of being a true positive, while at the same time learning that another type of vulnerability has a very low percentage of being a true positive. This can massively improve confidence of a true positive vs. a false positive. This is where automation can be increasingly helpful.

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

Management and monitoring 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. Automating KPI monitoring, tracking, and reporting as much as possible helps key stakeholders become and remain better informed overall.

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, and / or incentives. Teams can also determine if the policy in place needs refinement, etc. All of 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.

“

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.

Checkmarx

Conclusion

The whole point of this eBook was to foster a certain level of understanding pertaining to how to embed security into an organization's DevOps culture in the hope of helping organizations fully obtain what the industry calls ***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 the risks they face daily.

Integrating ***automation*** into as many areas as possible within DevSecOps is critical to improve quality, accuracy, security, and speed of delivered software. By following the guidance in this eBook, not only will organizations improve their security postures, but their KPIs and ROIs will increase over time as well. Faster, more-secure software releases are what many organizations strive for today and embedding security into DevOps, as demonstrated herein, will help them finally achieve their goals.

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 exposure.

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.

Best Fit for DevSecOps
Checkmarx leads the industry in delivering automated security scanning as part of the DevOps process.

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**