



# What the Heck is IAST?

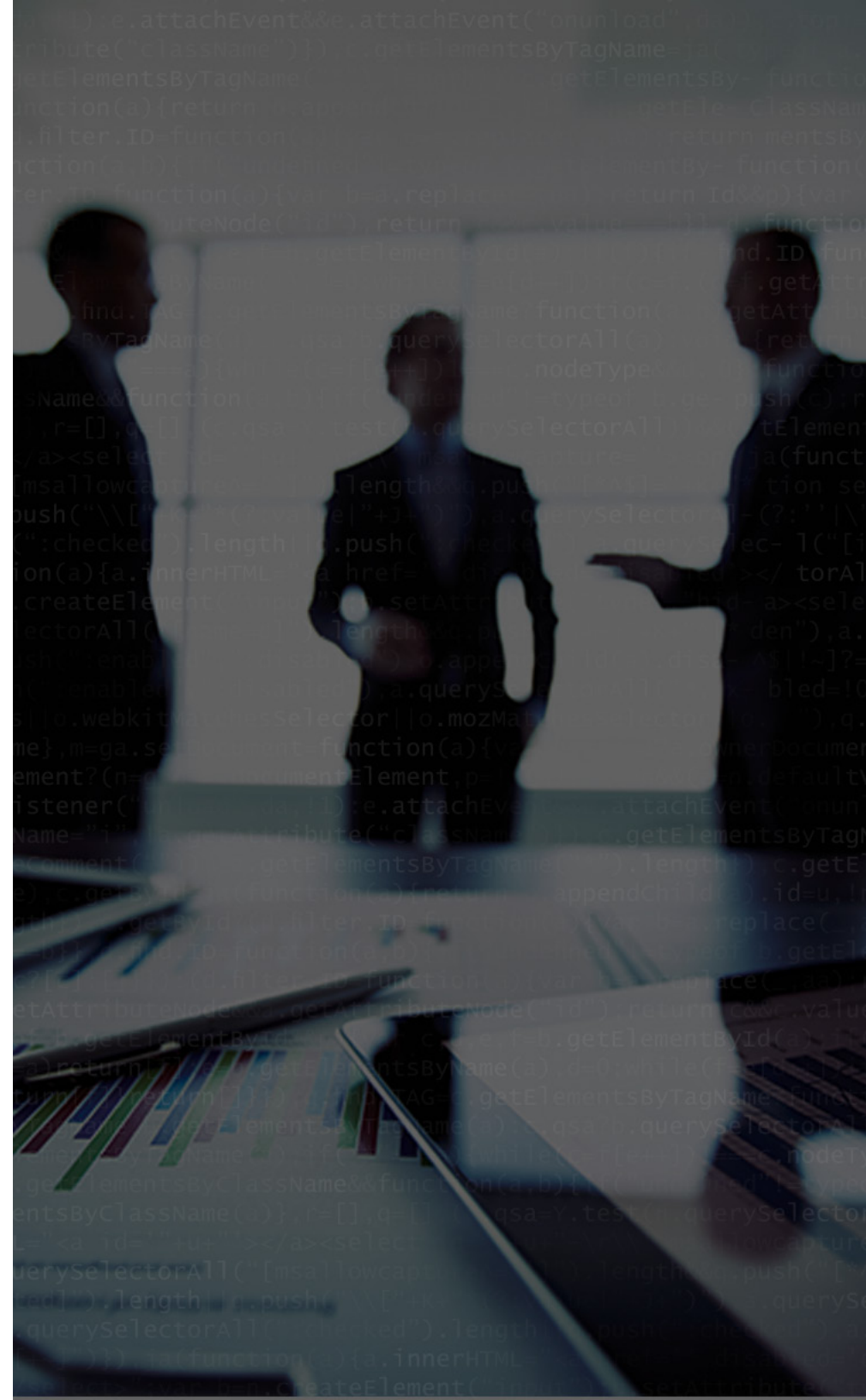


A Guide to Interactive Application  
Security Testing

Software = Security

# Introduction

In today's competitive world, time to market is imperative. Organizations are under increasing pressure to continuously deliver new and improved software. To win the race, nothing can get in the way of rapid software releases. This requirement has often led organizations to leave security behind, making them an easier target for a cyberattack. As awareness of the cyberthreat grows, and as the threat landscape continues to expand, organizations are increasingly seeking out software security solutions that support this highly iterative release frequency. As a result, Application Security Testing (AST) solutions designed to increase an organizations software security are in extremely high demand, and new technologies are emerging that exceed the capabilities of the available AST solutions in the market today. One of the new entrants into this market is called: ***Interactive Application Security Testing (IAST)***. This eBook discusses IAST at length.



# + Contents

INTRODUCTION	2
WHY THIS EBOOK	4
WHAT YOU WILL LEARN	5
<b>CHAPTER 1: Demystifying the Myth of an AppSec Silver Bullet</b>	<b>6</b>
- Why you need to scan your static code, and your running applications	7
- Why you need software composition analysis	8
- The secret to having your cake and eating it too	8
<b>CHAPTER 2: IAST vs. SAST and DAST</b>	<b>9</b>
- Different types of IAST	11
- How is IAST different from DAST	12
- How is IAST different from SAST	13
<b>CHAPTER 3: SAST &amp; IAST – The Power Couple in the DevSecOps Era</b>	<b>15</b>
- The traditional duo in the security gate era	16
- The new power couple in the DevSecOps era	17
POINTS TO CONSIDER WHEN BUYING AN IAST SOLUTION	18
WHAT YOU'VE LEARNED	20
CONCLUSION	21



## + Why This eBook

Nearly every statistic demonstrates that cyberattacks are growing beyond anyone's expectation. Year-after-year, product-after-product, solution-after-solution, organizations are exasperated trying to stay one step ahead of attackers, who would like nothing more than to steal an organization's sensitive data to be sold to highest bidder. Attend any information security tradeshow and you will see hundreds of solutions that try their best to defend against attackers, by relieving the symptoms of their attacks. However, very few of the available solutions you'll see can actually address the root cause of the situation – vulnerabilities in software itself. Today, organizations are looking to remedy this root cause.

Nearly every cyberattack that attempts to breach the confidentiality and / or integrity of an organization's data, focuses on exploiting vulnerabilities that were mistakenly made during software development. Unfortunately, those vulnerabilities found their way

into the fully-compiled running applications, making them ripe for exploitation. As a result, attackers exhaust vast amounts of their time trying to find these vulnerabilities in an organization's publicly-exposed software applications, then willfully exploit these vulnerabilities to their benefit.

Therefore, it's easy to recognize that the root cause of most 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 application security testing solutions, integrated into a single Software Security Platform, that address the root cause of nearly every successful cyberattack by finding, identifying, reporting, and demonstrating where-and-how to fix vulnerabilities in software.

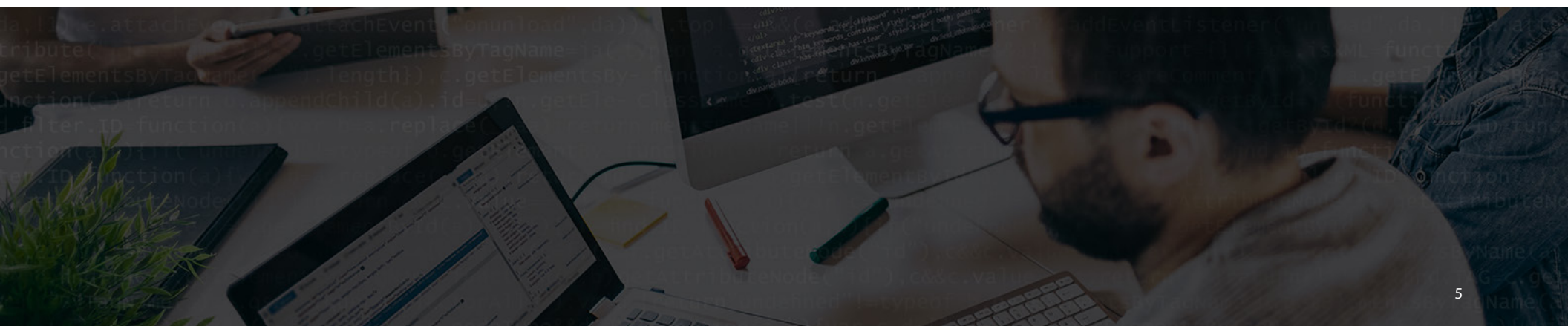
+ In this eBook, we will hone deeply into one of the industry's latest additions to the Application Security Testing (AST) marketplace, called **Interactive Application Security Testing (IAST)**. The reason for this eBook is to bring awareness to IAST solutions in general. Used during functional testing, IAST solutions are designed to help developers and security

personnel find vulnerabilities that could be exploited in their running applications – before attackers do. Understanding that the development of software applications is in a constant state of flux, IAST is designed to adapt quite well within the software development ecosystem.

## + What You Will Learn

In this eBook, you will learn what IAST is and why it's much different than the previously well-known products in the AST marketplace. We'll unravel the myth of an AppSec silver bullet, since there is no single, stand-alone application security testing solution available that guarantees the security of your software. We'll also discuss why you need to scan both your static code before compilation, and running applications during functional testing, while addressing the need for software composition analysis of third-party code in use within your applications.

Next, we'll discuss how IAST compares to other testing solutions and how it provides expansive coverage beyond what is currently available. We'll also discuss different types of IAST and how well they integrate within DevOps environments. Then, we'll delve into a new "power couple" in the DevSecOps era and how different testing solutions complement one another. Finally, we'll discuss points to consider when buying an IAST solution and list some business benefits to help you build your case for acquiring and implementing IAST in your own organization. After reading this eBook, you'll become much more knowledgeable of IAST overall.





## + Chapter 1

# Demystifying the Myth of an AppSec Silver Bullet



Can a single approach to application security solve all of your problems? Relying on a single testing solution to protect your applications is essentially like trying to protect your house from burglars, with a single alarm. Now imagine the house is packed with highly-valuable possessions and is located in a dangerous area – much like the conditions of your publicly-exposed applications. Few would feel confident that a single alarm would adequately protect them.

Similarly, there is no “silver bullet” when it comes to application security testing - any experienced AppSec professional knows that. To mitigate the risk associated with vulnerable software, you need to integrate security throughout your software development life cycle (SDLC). It's all about multiple layers, and multiple touchpoints. While it may seem obvious that no single application security testing solution can fully protect your applications, let's break it down to really understand why that is.

## + Why you need to scan your static code, and your running applications

Effective application security programs use testing solutions that scan code both *statically* during development, and *interactively* (dynamic) at run-time. Why are both of these testing types required? Firstly, because each one can identify different types of security vulnerabilities. For example, interactive testing is better at detecting deployment configuration flaws, while static testing finds SQL injection flaws more easily. Secondly, with the speed of today's development cycles — and the speed with which software changes and the threat landscape evolves — it would be foolish to assume that applications will be vulnerability-free after the development phase, or that code in run-time doesn't need to be tested. As a result, both types of testing are desperately needed, since no stand-alone testing solution can effectively perform both static and interactive testing.



As a result, both types of testing are desperately needed, since no stand-alone testing solution can effectively perform both static and interactive testing.

## + Why you need software composition analysis

Beyond static and interactive testing solutions, a software composition analysis solution is also required. Today's applications are essentially composed of hundreds of open source libraries that make up as much as 80% of the total code. With the speed of today's development cycles, developers don't have time to write every line of code from scratch, and why should they, when so much ready-made code is available. While there is no doubt that open source is the backbone of mass innovation, it is also opening a huge door for attackers. Every vulnerability in an open source library that your developers use is in fact, a vulnerability in your application(s). As such, it is not enough to just look at your proprietary code. Effective application security entails both assessment of your home-grown code, plus assessing and creating a dynamic inventory of your third-party code.

## + The secret to having your cake and eating it too


In order to have an effective software security program, it's clear you need to make use of different tools and solutions throughout your SDLC. However, this does not necessarily mean that you need to handle each tool separately from a different vendor. When it comes to AppSec solutions, the real secret to having your cake and eating it too is to find a vendor that offers a completely integrated platform that performs static, interactive, and open source testing. With a single integrated software security platform, you can gain more than you would with multiple-silo solutions. With an integrated platform, not only will you benefit from a single support point, reduced TCO, and better ROI, but you will also gain from the product synergy that comes with an integrated platform, such as unified policies, results correlations, and centralized management.

Having understood that no one AST solution can guarantee the security of your software, we now go into the differences between three major testing solutions – IAST, SAST, and DAST.



With an integrated platform, not only will you benefit from a single support point, reduced TCO, and better ROI, but you will also gain from the product synergy that comes with an integrated platform, such as unified policies, results correlations, and centralized management.



A man and two women are gathered around a laptop in a server room. The man, wearing a headset, is pointing at the screen. The woman on the left is also looking at the screen, while the woman on the right is looking down at a document. The background is filled with server racks and blue lighting. Overlaid on the image is a large, semi-transparent text box containing the title and chapter information.

## + Chapter 2

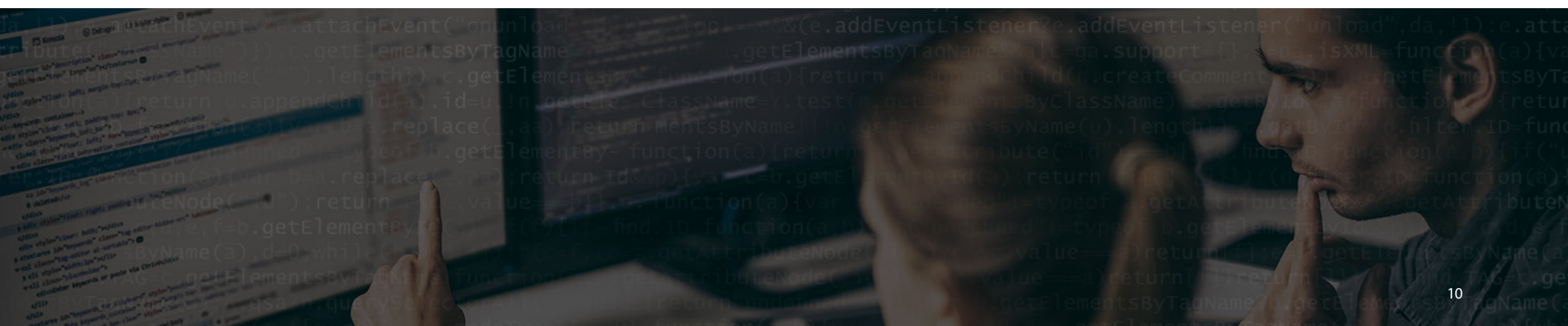
# IAST vs. SAST and DAST

The application security testing (AST) world is made up of different solutions, all with one ultimate goal – to detect and help developers remedy code defects that lead to vulnerabilities. This ultimately protects software applications from attackers, and their attacks. SAST and DAST are perhaps the two most common and well-known solutions. In the last few years, a newcomer has gradually received more-and-more attention – IAST.

Analyst firm Gartner defines IAST as follows: “Interactive application security testing (IAST) uses instrumentation that combines dynamic application security testing (DAST) and static analysis (application) security testing (SAST) techniques to increase the accuracy of application security testing. Instrumentation allows DAST-like

confirmation of exploit success and SAST-like coverage of the application code, and in some cases, allows security self-testing during general application testing. IAST can be run stand-alone, or as part of a larger AST suite, typically DAST.”

Gartner’s definition is quite broad, allowing for a variety of solutions to be classified as IAST products. To really define what IAST is, we need to make a distinction between two types of IAST – Active vs. Passive. Both active and passive IAST rely on an agent instrumented within the application itself during the testing stages of the SDLC. However, the differences between the two methodologies are significant in terms of the technology itself, in addition to how well they fit into fast-paced DevOps environments.





## + Different types of IAST

Active IAST, also known as “Induced IAST”, requires a DAST solution in order to provide any real value. By instrumenting the application with an agent, the DAST engine will run its attack simulation and depend on a response from IAST to validate an existing vulnerability. While this approach provides results with great accuracy, it requires its own testing environment and cannot be automated. Therefore, this approach is viewed as unfit for fast-paced CI/CD and DevOps environments.

Passive IAST, or more correctly termed “Self-sufficient IAST”, was actually built to overcome the caveats of active IAST – by addressing the need for quick application security testing in fast-paced DevOps environments. Passive IAST also requires instrumentation of the application in the testing environment. But rather than actively running dedicated tests or attacks, it will leverage any form of functional testing to collect data and deliver accurate security findings in zero time. Whether tests are automatic or manual, the passive IAST will “listen in” and continuously report on any findings.

In Figure 1 below, a summary of the differences between Active and Passive IAST is shown.

Figure 1



### Active IAST

- Requires a DAST solution in order to provide any value
- Provides results with great accuracy
- Requires its own testing environment
- Cannot be automated



### Passive IAST

- Leverages any form of functional testing to collect data
- Delivers accurate findings in real-time
- Uses the existing testing environment
- Can be fully automated



## + How is IAST different from DAST

DAST tools find vulnerabilities on running applications by externally attacking the application. Coverage is limited to reflective types of vulnerabilities, since DAST solutions are essentially blind as to what is happening inside an application. Some of the challenges experienced with DAST include moderate false-positive rates, an increased number of testing cycles, and increased testing duration. Finally, DAST 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 for integration into stringent CI/CD workflows, unlike IAST, which produces results in real-time.

DAST is often considered as “black box security testing”. DAST tests running applications from the outside-in and simulates attackers

running their various attack scenarios, similar to penetration testing. However, DAST has no understanding of the internal details of your code. Observing external output of the application, while injecting input to external interfaces, DAST requires a dedicated security testing environment and is usually not incorporated into developers' integrated development environments (IDEs). The main drawback of DAST is that it cannot detect non-reflective vulnerabilities, for example XSS (Cross-Site Scripting). Due to its operational overhead, in most cases DAST does not fit well into DevOps, since it can introduce delays in Agile and CI/CD workflows.



The main drawback of DAST is that it cannot detect non-reflective vulnerabilities, for example XSS (Cross-Site Scripting). Due to its operational overhead, in most cases DAST does not fit well into DevOps, since it can introduce delays in Agile and CI/CD workflows.

## + How is IAST different from SAST

The main difference between the two pertains to the name – Static vs. Interactive. SAST, which stands for Static Application Security Testing, analyzes the application source code to identify vulnerabilities. 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. However, unlike IAST, SAST is blind to how all the pieces of an application work together and operate at runtime, so it can't detect vulnerabilities in running applications that attackers may be able to exploit. In comparison, IAST performs its testing during run-time, being capable of detecting vulnerabilities in running applications.

SAST is often considered as “white box security testing”. These technologies test from the inside of the code itself. SAST understands all of the internal details of your code and it's a perfect testing approach for developers and security teams. Scanning code or binaries without executing the application, SAST does its scanning before an application

is compiled. SAST scans and detects vulnerabilities in your in-house code, but is incapable of detecting vulnerabilities in third-party or open source modules. A tremendous value of SAST is that it doesn't delay development cycles and doesn't require scanning the entire code base, since it's fully capable of supporting incremental scans.

Finally, IAST is often considered as “grey box testing” which is indicative of having similar attributes of both SAST and DAST. IAST is somewhat of a hybrid of both testing approaches. Testing from both inside-out, and outside-in, IAST can understand some of the internal details of the applications' code itself. IAST monitors and analyzes the applications, but doesn't require testing on code or binaries to detect vulnerabilities. Its coverage is as wide as the testing, since it's always active, waiting to analyze the application's activity during functional testing. IAST also has the ability to cover third-party modules and its' fast and immediate results don't impact or delay the CI/CD process.



IAST also has the ability to cover third-party modules and its' fast and immediate results don't impact or delay the CI/CD process.

In Figure 2 below, a summary of the differences between SAST, IAST, and DAST is provided.

Figure 2



Having understood the difference between these three AST technologies, next we examine how the use of these solutions has changed over the years to fit with the fast pace of release. Leaving DAST out of the release cycle, we now see the need for a strong integration between static and run-time testing solutions as part of the release cycles.





## + Chapter 3

# SAST & IAST – The Power Couple in the DevSecOps Era

DevSecOps has become one of the hottest buzzwords in the DevOps and security ecosystem over the past couple of years. But what is it, and how do you turn it into reality?

DevSecOps executes on the belief that security and development teams are jointly responsible for bolstering security – essentially bringing development and operations together. This methodology “bakes” security in as early as possible, covering the entire SDLC, with the aim to find, fix, and prevent security vulnerabilities without degrading productivity or time-to-market.

In theory, it’s easy to understand what DevSecOps means and why people care about it. But practically speaking, how do you actually achieve it? The reality is that many organizations that are adopting Agile and DevOps methodologies discover that the security tools they once used can no longer keep up with the speed and frequency of releases.

Here we explore the traditional SAST-DAST AppSec approach, the challenge it is facing when it comes to today’s pace of release delivery, and we introduce the new power couple to fit the DevSecOps era – SAST and IAST.

## + The traditional duo in the security gate era

Traditionally, and to remain secure, organizations implemented AppSec programs that were primarily characterized by the use of toolsets that analyze the code or binary itself (i.e., SAST) or assisted operators in performing simulated attacks to see how an application would react (i.e., DAST). These testing tools were typically managed by a team of security experts who operated in isolation and often requested developers to change the way they worked.

Fast forward to today, multi-day static and dynamic analysis run by a small pool of security experts is not a tenable model, when the business demands multiple software releases per day. While SAST tools can be pushed beyond security usage to the developers, DAST tools cannot. DAST tools must be operated by experienced AppSec teams to truly be useful, making them unsuitable for environments that foster automation and fast turnarounds.



The reality is that many organizations that are adopting Agile and DevOps methodologies discover that the security tools they once used can no longer keep up with the speed and frequency of releases.



## + The new power couple in the DevSecOps era

We know that in today's competitive world, the name of the game is time-to-market. Organizations are under growing pressure to continuously deliver new and improved software. To outpace your competitors, nothing can get in the way of rapid releases. If security is not to be compromised, testing tools that can be automated and integrated into your SDLC are a requirement.

When it comes to scanning your code, you need a SAST solution that can integrate with your CI/CD pipeline and deliver rapid, consistent results with low false-positive rates. To truly scan your code in an agile manner, a solution that provides incremental scanning is required. If you wait until the end of the SDLC to run a full scan of a built application, it will take you more time, attention, and will ultimately cost more money to resolve coding issues.

Similarly, when it comes to securing your run-time applications, you can't afford to wait for a few days to get DAST results. You need a solution that can be integrated with your CI/CD pipeline

and automated as part of every release. This is where IAST solutions come in. Unlike legacy DAST tools, IAST provides real-time vulnerability detection and immediate feedback. Developers receive security feedback as soon as they run their code, so there's no need to wait for additional scan processes to finish. At the same time, QA testers can quickly identify security vulnerabilities without extensive application security experience.

In summary, the real key to successfully adopting security solutions that fit DevSecOps is integration. While it's great to have SAST that can incrementally scan your code, and IAST that protects your runtime application in real-time, the real DevSecOps benefit comes from bringing the power of these two together.

At the end of the day, what's the use in automating vulnerability scanning and making it faster, if you'll still be slowed down by the need to aggregate the different results and make sense of it all. So remember, when evaluating SAST and IAST solutions, it's important to understand how easy it would be to correlate between the two.

+ When organizations are considering the purchase of an IAST solution, there are several points that must be considered. These points are discussed in the next section.



## + Points to Consider When Buying an IAST Solution

Although there are many options when moving forward with an IAST purchase, the following should be part of the decision-making process. Not all IAST products on the market today provide the exact same functionality and some limitations can be observed between different solutions.

### Does the IAST solution require an inducer?

As previously mentioned, there are two types of IAST available (Active vs. Passive). An Active IAST requires an **inducer** to simulate attacks from the **outside-in** on a running application. In many cases, the inducer would be a DAST scanner attacking the application while an IAST agent is running on the target server. Although this approach provides broader scanning results, it's also considerably slower in finding vulnerabilities. In addition, the inducer must be maintained and updated when changes are made to the application's UI, logic, authentication methods, etc. Finally, depending on the type of

testing being performed in this approach, it can be potentially destructive and requires its own testing environment.

On the other hand, Passive IAST does not require an inducer. Instead, it takes advantage of the normal functional testing environment, tools, and processes. This approach also has an IAST agent running on the tested application server with the agent listening in the background, discovering security vulnerabilities during functional test execution. The functional testing is exactly the same as before installing the IAST agent and does not have to be security aware, since this method does not attack the application but looks for vulnerabilities during normal application usage. Another benefit is that organizations don't have to train their functional testers, QA people, developers, etc. to be security experts, since it leverages existing testing processes. Ensure you know what **type** of IAST you're considering to purchase.



Not all IAST products on the market today provide the exact same functionality and some limitations can be observed between different solutions.

### Can IAST results be correlated with SAST results, providing developers with actionable Info?

Many organizations already use a SAST solution. What organizations would like to experience from IAST is the ability to correlate its test results with the results from their SAST solution, providing faster time to remediation with a complete view (source and running application data) of the vulnerability. This enables developers to find the “best-fix-location” quickly to remediate related vulnerabilities, as well as understand how a user would execute the code in a running application. Ensure the IAST you’re considering to purchase provides results that can be directly correlated with your SAST results.

### What automation capabilities does the IAST provide?

Active IAST is typically difficult to automate. On the other hand, Passive IAST can easily be inserted into the CI/CD pipeline in an automated fashion. This automation ensures that deployment processes are not impacted or delayed, waiting for tests to complete. Passive IAST is

always running during functional testing and produces results in real-time. Ensure the IAST you’re considering to purchase can be easily automated into your CI/CD processes.

### What are the detection and false positive / false negative rates of the IAST?

Vulnerability detection rates and validity of results are of upmost importance for any AST tool. False positives (FPs) normally create lots of noise and often produce **alert fatigue**, causing reduced attention to test results. In comparison, false negatives (FNs) are of even greater concern, since they mean a vulnerability does exist, but was not detected. The promise of IAST is that the FP rate is much lower than seen in SAST. However, a small number of FPs and FNs may be experienced. Therefore, determine how easy is to address them by extending or adjusting the IAST queries or rules. Ensure the IAST you’re considering to purchase has extremely low FP and FN rates and supports flexible and configurable queries and / or rules.



This enables developers to find the “best-fix-location” quickly to remediate related vulnerabilities, as well as understand how a user would execute the code in a running application.

## What overhead or delay does IAST introduce into testing?

IAST must fit your business needs and objectives. Finding security vulnerabilities in running applications as part of your existing pipeline cannot impact your objective of deploying an application on schedule. Passive IAST results are provided in real-time, whereby Active IAST results are provided after the fact. Ensure the IAST you're considering to purchase provides results with absolutely no delays.

The case for needing Passive IAST is quite strong and the points above tend to minimize the value of an Active IAST approach. Finally, Passive IAST offers many more touchpoints for developers and security personnel during the entire SDLC, and leverages the automated or manual functional testing of the application normally performed anyway.



The case for needing Passive IAST is quite strong and the points above tend to minimize the value of an Active IAST approach.

## What You've Learned

In this eBook, the intention was to introduce you to IAST overall, and many topics were discussed throughout. First, we demystified the myth of an AppSec silver bullet, meaning, there is no single solution in the AST marketplace today that can provide complete coverage on its own. Next, we discussed the differences between Active and Passive IAST and how IAST operates with both DAST and SAST solutions.

Then we discussed the traditional duo in application security testing solutions and highlighted how IAST and SAST together, provide greater security vulnerability coverage than the traditional approach. Finally, we highlighted some points to consider when buying an IAST solution. After reading this eBook, you should have a sound understanding of all the concepts and topics discussed herein, concerning IAST overall.



# + Conclusion

Reducing development time-to-market is the new standard. If an organization can provide new features faster than its competition, it will gain market share. This has led to widespread adoption of DevOps teams using agile development methodologies to deliver new features faster. This also brings new challenges to security teams trying to adjust to this new environment.

Checkmarx delivers a Software Security Platform for all aspects of DevOps including development and CI/CD environments by redefining security's role in the SDLC, all while operating at the speed of DevOps. The fast feedback loop makes security testing of new or edited code fragments quick, allowing speedy remediation by developers. This significantly reduces costs and eliminates the problem of having to deal with many security vulnerabilities close to release.

Today, IAST solutions allow security to be an integral part of your SDLC. Unlike DAST, IAST is designed to eliminate delays in finding vulnerabilities during functional testing and does not slow down the DevOps cycles. Since it has been proven herein that IAST is the perfect complement to SAST, organizations who take advantage of the inherent product synergies will gain measurable business benefits, produce more-secure code, and significantly reduce their overall risk profiles. Simply put, IAST is the only dynamic solution that can fit DevOps, within the Application Security Testing marketplace.



# + 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.

If you would like to learn more about how your organization can integrate CxIAST into your DevOps and software security initiatives, please request a live demonstration at [www.checkmarx.com/products/interactive-application-security-testing](https://www.checkmarx.com/products/interactive-application-security-testing)

Software = Security