SYNOPSYS®

WHITE PAPER

Solving the AppSec Puzzle:

Connecting AppSec to Your DevOps Pipeline

Introduction

Integrating application security (AppSec)—the technologies used to detect and address potential security issues in software—into your software development life cycle (SDLC) and your DevOps pipeline is increasingly important in a development environment characterized by continuous integration/continuous development (CI/CD) workflows. AppSec integration allows security teams to establish security gates at multiple stages of the development and deployment process, which helps you avoid late-stage testing and the development rework that ensues. Late-stage testing and development can delay releases or lead to overlooked risks being promoted into production. This approach is commonly referred to as "shifting left" in the development cycle or, increasingly, as "shifting everywhere."

True AppSec integration requires combining and connecting elements, systems, and processes to work together seamlessly. It involves merging disparate components into a unified system, enabling the smooth flow of information, functionalities, or resources. Moreover, integration means these elements and systems fit naturally into established workflows—preventing teams from architecting new workflows that disrupt teams and diminish efficiency.

In the context of software development and application security, integration plays a crucial role in achieving important business objectives, including

- Mitigating risks that could threaten sensitive data
- Supporting compliance initiatives for security practices
- Elevating efficiency standards during development, testing, and deployment

AppSec integration extracts information and delivers risk insight

AppSec integrations make it possible to extract valuable information from various stages in the pipeline and enable you to deliver risk insight directly into developer workflows at the point where they can mitigate those risks as they continue their development work. This helps organizations establish automated processes that accelerate risk detection and prioritization, and prevent issues from proliferating downstream—all without risking missing a software shipping deadline.

AppSec integrations capture and extract data from multiple sources including development tools, code and binary repositories, version control systems, build systems, testing environments, and production environments. These integrations also allow organizations to run the right tests, at the right time, and at the right depth. This means security teams are not constrained to a single tool or testing protocol at a time. Rather, relevant tests run at various stages of the DevOps pipeline mitigate pipeline congestion.

Additionally, such integrations help teams catch vulnerabilities that may have been introduced at any stage of the development process, such as a developer checking a third-party binary or component into a repository without performing early-stage testing. Using integrations enables you to add a level of redundancy into your security testing at each stage, which helps ensure that nothing is missed.

AppSec integration also allows organizations to deliver risk insights at multiple points across the pipeline. This type of integration means security-related data such as code scan results, security event logs, and vulnerability alerts is disseminated efficiently and aligned to organizational standards for risk tolerance. Getting this information into the hands of those who can use it to fix associated security issues enables organizations to propagate informed decisions about risk prioritization and mitigation strategies, without being limited by subjective assessments of risk or inadequate security capabilities among developers and DevOps teams.

These two concepts—assessing risk and delivering actionable risk intelligence across the SDLC and DevOps pipelines—are fundamental to elevating AppSec programs to achieve DevSecOps. As software development and deployment become more interconnected, AppSec testing must become an intrinsic part of this framework rather than an appendage.

Security as a business driver

Security is essential to any business because, at the very least, it keeps you from running afoul of regulatory guidelines or sacrificing sensitive data. But too often, these basic functions relegate security programs to the status of "cost center." As businesses in every industry and sector become increasingly defined by software, secure software development and DevSecOps must shift from something organizations see as costly protection to a potential business driver.

This means integrated application security that drives reputational and customer trust can garner a market advantage. Organizations that prioritize security and demonstrate a strong commitment to protecting customer data enhance their reputation, gain customer trust, and potentially attract new customers. Customers prefer to engage with vendors that demonstrate robust security practices and have a proven track record of safeguarding customer information. Organizations can also gain a competitive edge, as customers are more likely to choose secure vendors over those with poor security practices.

Partnerships and supply chain relationships rely heavily on attestation of security and proof of automated safeguards integrated into products and systems. In many cases, businesses are required to meet certain security standards or demonstrate compliance with security frameworks to collaborate with partners or access specific markets. Ultimately, enterprises do not want their risk posture to depend on the security inadequacies of partners and vendors.

Organizations that demonstrate clearly defined practices for security testing and DevSecOps maximize their business potential through consistent, efficient operations that ensure revenue-generating web properties or applications that continue to perform. Investments in security pay themselves back by protecting your customers, your data, your reputation, and your ability to keep your business up and running in an environment where business never sleeps.

Complex software complicating security

Software has become an integral part of our daily lives, from the applications on our smartphones to the systems that power businesses, industries, and governments. Software is no longer monolithic, but is instead composed of many types of components tied together by protocols and systems that allow for unified execution of operations and data transmission.

Your business runs on software you write, the software you borrow, and the software you purchase. The first is constructed by your developers to meet the specific needs of your organization, and the second is composed of assets your developers leverage to accelerate development. It includes open source software and third-party libraries that can be used, modified, and distributed per license permissions. Lastly, there are third-party licensed software binaries that organizations acquire from external vendors, and through the software supply chain, to integrate into their projects and systems.

Each of these software classifications may be constructed atop diverse frameworks and technologies that turn code and components into running programs. This includes containers to compartmentalize aspects of software functions, microservices for agile scalability, infrastructure-as-code (IaC) templates to define cloud resource configurations, API protocols, and more.

With each additional technology, software security becomes more complicated, and maintaining high throughput and rapid release cycles becomes a more delicate balance.

Defend against diverse threats

Whether it is built in-house or consumed from external sources, software must be secure to protect sensitive data, ensure privacy, and prevent unauthorized access or malicious activities.

All this software being developed, borrowed, and bought means that there are corresponding opportunities for security vulnerabilities and exploitable weaknesses to wind up in your code. The interconnectedness of all these kinds of software means that a security risk in one component can have a cascading effect on an entire system.

As your software composition becomes more complex, so too must your testing technologies and security practices. Detecting weak or insecure coding practices in proprietary code, identifying known vulnerabilities in open source components, evaluating the security of data transmissions, and observing potentially malicious activity in running compiled assets each poses a unique challenge to an application's overall security posture.

To fully achieve DevSecOps, technologies need to be integrated into established workflows to trigger relevant testing, avoid unnecessary testing that could delay software shipment, and automate enforcement of risk-based policies.

Challenges to building robust DevSecOps

There are a number of challenges to integrating security controls and practices into DevOps systems in order to build truly secure DevSecOps workflows.

Ensure application accessibility

One challenge that organizations face when trying to integrate AppSec tools across the pipeline is that no two software components are exactly the same, even if they perform similar functions. Different programming languages, frameworks, libraries, and versions can introduce variations in how software components can be secured. This is an important consideration when it comes to security testing, as testing tools need to account for these distinctions to effectively identify vulnerabilities and security weaknesses.

Security profiles differ considerably for external and internal applications.

- **External applications:** Internet-accessible or external applications are traditionally more at risk from malicious attacks and unlawful access. You need to test for vulnerability to injection attacks, XSS, CSRF, authentication bypass, and other web application security threats to external applications. Furthermore, externally accessible applications tend to be very sensitive to tests or conditions that may consume cloud resources and slow performance. These applications also tend to collect sensitive information from users outside a protected network and who often have no additional security safeguards in place.

  It is paramount that security teams be comprehensive when analyzing external applications for security risks, as there is increased exposure to a population of malicious actors of all skills and specialties. Security and DevOps teams must enact intelligent integrated security practices to preclude issues without impacting release cycles or performance of public applications.

- **Protected internal applications:** Internally accessible applications are traditionally meant to be used and installed on protected internal servers or network segments. These applications may not be accessible from the internet, or may require additional permissions and credentials to gain access. While this can limit potential attack vectors, it can also inspire complacency with access-based security and overlook integrated DevSecOps to harden the application.

  The highly sensitive nature of the information that internal applications access is an attraction for sophisticated attackers. Strong security measures must exist to prevent insider threats and lateral movement by attackers who might acquire illegal access to internal networks. Additionally, modern security teams require integrated security testing that closely examines software in the pipeline for potential data leakage and that documents vendor and third-party assets in the event of a compromised supply chain.

To effectively address the distinct security profiles of external and internal applications, you should combine automated security testing methods, such as vulnerability scanning and static code analysis, with manual penetration testing, code reviews, automated preproduction functional and security testing, and lightweight production security testing.

Overcome DevOps pipeline complexity and variance

The complexity of DevOps pipelines and the lack of a uniform process, even within organizations, poses significant challenges to engendering end-to-end application security practices. Organizations often establish DevOps pipelines that follow standard processes from development, passing through security checks and quality controls, performing functional testing, and then into production.

Developers, however, may resort to unapproved parallel pipelines or custom workflows to expedite development or foster experimentation. These side pipelines might bypass certain security measures or introduce unapproved components, making it difficult to have visibility into the security posture of assets that pass through these pipelines or maintain control over their implementation. This lack of visibility and control can increase the risk of vulnerabilities or misconfigurations slipping through undetected.

The software supply chain is an important example of the need to integrate security into DevOps, as software components from various sources, including open source libraries, third-party dependencies, and container images, can include vulnerabilities or malicious artifacts. There is additional complexity and risk with the software supply chain, as many elements of leveraging these assets are automated. For example, a development team may make calls to pull in the latest version of an open source component without regard for its vulnerability status, an unknown (transitive) dependency may introduce vulnerable or malicious software during a build, or a compromised software vendor may transform previously trusted binaries into clandestine attack vectors.

Integrating security measures into your supply chain is crucial to ensure that components are vetted, examined for vulnerabilities or malicious activity at runtime, and that security and development personnel are notified immediately to any changes in security risk posture.

The variety of distinct tools and workflows that developers rely on to accomplish their work can further complicate matters for security. Security tools may offer only partial coverage for the specific programming languages, frameworks, IDEs, or build tools developers use. No single security tool can provide complete coverage across the diverse pipelines and technologies used in DevOps. As such, organizations need to leverage a combination of security tools that allow for comprehensive security testing, vulnerability scanning, configuration checks, and code analysis, and do so in such a way that minimizes unnecessarily duplicative efforts or delays.

Manage distributed development and testing

Distributed development and testing pose significant challenges to integrating security into the DevOps pipeline. In distributed development environments, different teams or business units have their own preferences, priorities, and budgets when it comes to choosing tools and technologies. This can result in a diverse tooling landscape across the organization, making it challenging to integrate and align security practices. Redundancy and inconsistency can arise when teams use different security tools, and that can lead to gaps in coverage and unnecessarily duplicated effort.

Distributed or nonlocal development teams work in different locations and time zones, making it challenging to enforce consistent—and timely—security processes and standards. The lack of physical proximity can hinder effective communication, collaboration, and real-time coordination on security-related issues. This nonlocality can also mean that teams are not using local development and DevOps tools on secured networks. Instead, they may tend toward cloud-based, as-a-service pipelines that can make establishing security controls more difficult if the security tools lack direct integration into cloud-based platforms.

Meet operational and organizational pressures

Operational pressures also present significant challenges to integration in the DevOps pipeline. Projects may have varying development, release, and testing cycles. Some projects may follow weekly sprints, while others may have quarterly or even longer update cycles. These different cycles can make it difficult to establish consistent security practices unless you directly integrate controls and triggers into the SDLC, DevOps workflows, and CI/CD pipelines. As development and DevOps teams shift toward more-rapid and continuous release cycles, integrating and aligning security becomes imperative. This is true not just for detecting risks, but for supporting more-effective remediation. When developers are forced to drop current work to remediate risks from past sprints, your current deadlines can be affected.

Meanwhile, the demand for thorough security testing and expertise often exceeds the available resources within organizations. This can be especially challenging in DevOps, where the need for continuous security assessments and monitoring is essential, and evolving threats require investment in evolving security solutions. Similarly, top-down organizational mandates may lack awareness of the security and workflow pressures experienced in business units, departments, and development teams distributed around the globe.

Lastly, enterprises and distributed workforces often foster sprawling portfolios of security tools from multiple vendors. These may be redundant or even contradictory, leading to wasted effort and funds. These tools each have different integration and performance capabilities and can lead to unpredictable results as security responsibilities crossover between groups and risk management coalesces for efficiency.

Security integration adoption strategies

There are a number of strategies you can use to address—or prevent—challenges to integrating security in your DevOps workflows.

Intelligently coordinate multiple testing methodologies

To address all the software components and processes in the DevOps pipeline, organizations should employ multiple testing methodologies. We mentioned many of these in earlier sections, including

- **Static application security testing (SAST):** This manner of security testing examines proprietary code written by developers for common weaknesses that may be exploitable under certain conditions.
- **Software composition analysis (SCA):** This manner of testing detects and identifies known open source components and libraries within applications during development and after the build. This includes unknown transitive dependencies that are introduced into the application during the build.
- **Interactive application security testing (IAST):** This manner of testing effectively turns preproduction functional testing into security tests by running alongside preproduction runtime applications and monitoring data access, transmission, and other mechanisms that can lead to sensitive data leakage if configured insecurely. This type of testing is often aligned to compliance and regulatory guidelines for data security, such as PCI DSS and GDPR.

There are other testing methodologies, but these three comprise an efficient and effective way to detect and identify the risks present within software from the beginning, and that may enter through the SDLC and CI/CD pipelines (except for malicious software, of course, which may be free of known vulnerabilities and weaknesses, but that may represent a threat in production runtime or at some point in the future as determined by attackers).

It is important to intelligently coordinate these tools at their appropriate location in DevOps workflows to ensure the most comprehensive and least disruptive coverage possible. This means running the right tests at the right time, and running the appropriate tool for the software being tested and the change that has been made to it. For example, if a developer only changed an open source component version, there's no reason to run SAST. If they wrote new proprietary code with no open source, there's no reason to run SCA. If you triggered a test at code commit or build, it would be nonsensical to run IAST without a running application. But this type of coordination relies heavily on policies defined by security teams and aligned to the workflows and technologies used by development and DevOps teams.

Use policies and automation to establish security gates

Many SAST, SCA, and IAST solutions provide the capability to set policies that establish risk tolerance thresholds or required activities, although it is important to know which mechanisms each policy can support (e.g., pipeline activity, code changes, risk metrics) and what automated action may be taken upon violation of such a policy. These policies must be fully integrated with the tools and systems used by each contributor, from development into production.

It is also important not to make policies so granular that they apply to an irrelevantly small sample set of applications, or so broad that they generate distracting noise and alerts. They should be aligned to the success criteria of each team, but centrally managed by the security team to ensure no deviation over time.

Extract risk data across your pipeline, centralize analysis for prioritization

Properly integrated security gathers data from various points across the DevOps pipeline with efficient and early testing. For instance, you can trigger tests as new development work is performed or code changes are made. By scanning the code at this stage, developers can receive immediate feedback on potential security vulnerabilities they may have introduced with open source components, or insecure coding practices they may have employed during development. This shifts security as far left as possible and helps address issues before pushing code downstream.

At the build or CI stage, you can automatically detect vulnerabilities introduced during a build or pulled in from internal or external repositories without having passed through earlier security testing. Combining security testing with functional testing allows you to integrate security in a way that generates additional risk insight without additional effort.

It is important to leverage a centralized tool and security risk insight management system that can collect, normalize, cleanse, and prioritize the information it ingests. This requires a tool that can integrate directly with security testing solutions from various vendors, assess the data through centralized policies, and generate a clean list of prioritized results and artifacts that violate standards for risk tolerance. This allows a more uniform assessment of risk, regardless of the distributed nature of the development and DevOps organizations and the breadth and complexity of the tools they use. Once security test results are correlated and triaged, clear alerts and remediation guidance can be automatically shared with development teams.

Deliver security risk insight directly to developer workflows

All the extraction testing in the world isn't going to help if you can't deliver the risk intelligence and remediation guidance to the developers who fix issues in the software. In cases when integrated security testing occurs early in the development pipeline, it's important to also propagate this information to the security team for tracking, so it can check the remediation solution at the security testing stage.

This risk insight needs to be clean, relevant, and prioritized to ensure that any follow-on activity supports security standards as well as maintains optimal DevOps velocities. It should clearly define these specific details.

- **Risk identification:** This includes a detailed description of the issue, the context in which it manifests in the application, and where in the code or file structure it exists. This information may include a standardized identifier of the risk, such as a common weakness enumerator (CWE) or common vulnerability enumerator (CVE), and should ensure that a developer has adequate context to understand the issue.
- **Risk assessment:** This includes any standardized metrics of risk, such as the common vulnerability scoring system (CVSS) rating, to help prioritize risks for remediation. Often, these standardized metrics are incorporated into policies due to their objective nature. Custom scoring that's relevant to a particular organization may also be included, if defined.
- **Remediation guidance:** This information may be the most subjective and difficult to obtain, unless you have an internal team of cybersecurity experts to make recommendations, or if you have the support of a third-party knowledgebase or cybersecurity team. Such guidance should account for the diverse skill levels of developers and provide clear, actionable recommendations for how to fix a detected issue. This may even include links to associated eLearning to educate developers and help them prevent issues at the time of writing code in the future.

Risk insight helps contributors play a more effective and efficient role in upholding security standards for software that passes through DevOps pipelines. Once the insight is cleansed and prioritized for remediation based on integrated policies, it must be disseminated to each team via the tools and systems they use as part of their daily workflows.

- **IDEs and issue-trackers:** Providing developers with notifications, alerts, or annotations within their IDEs enables them to address security vulnerabilities promptly and in context. This can be enabled through direct integrations between security tools and IDEs, through APIs, or through native plugins for specific IDEs that are created by software security vendors. Additionally, security risk insight can be delivered directly into issue-trackers, ensuring that security concerns are tracked and prioritized alongside other development tasks. Such integration also provides a benefit to security teams, as bidirectional feedback capabilities can notify security teams of issue resolution.
- **Repositories, registries, and reports:** After risk identification and review by security teams, software can either be modified by developers to fix issues or acknowledged as compliant with security risk tolerance and promoted through the pipeline. But security status is only a temporal snapshot of risk and can change as new vulnerabilities, exploits, or attack methodologies emerge. This will occur after software is put into production, where operations or DevOps teams bear responsibility for knowing the security status of software. As such, these teams will benefit from a Software Bill of Materials to monitor for new vulnerabilities, risk reports, logs, or container registry annotations that serve as an attestation of risk.

Integrating risk insight across your organization means that your stakeholders will have access to timely information across the DevOps pipeline. Developers who receive immediate feedback can address security issues during the development process. The security team remains aware of potential vulnerabilities, allowing them to take proactive steps to protect the organization. DevOps or operations teams stay informed about new vulnerabilities that may impact production environments, enabling them to maintain the security and stability of deployed software. And security and technology leadership can stay aware and involved in maintaining security standards and compliance with risk tolerance thresholds. Overall, integrated risk insight enhances collaboration, accelerates remediation, and promotes a culture of security awareness and security capability throughout the DevOps life cycle.

Recommendations for realizing integrated DevSecOps

To establish an effective, integrated DevSecOps program, it is important to ensure that you have the proper security tools and configurations in place. There are a number of application security steps and tools that can implement those steps and fully integrate security risk awareness and security gates across the SDLC and CI/CD pipelines.

Establish security in the IDE

Integrating SAST and SCA into developer IDEs can be done using a solution like Synopsys Code Sight™, a plugin that helps developers identify security risks in their proprietary code and open source components without leaving their preferred IDE (e.g., VS Code, Eclipse, IntelliJ). Code Sight provides detailed remediation guidance from the Synopsys Cybersecurity Research Center (CyRC) to educate developers on how to address and avoid security risks. This allows them to find and fix issues before pushing code downstream and helps them avoid unnecessary rework due to failing late-stage security testing.

Synopsys Code Sight can function as a standalone tool that helps development teams become more security-aware and security-capable and it can extend the comprehensive capabilities of Coverity® SAST and Black Duck SCA, which may be wholly managed by security teams. This provides real-time notification of policy violations directly within the IDE, as well as IDE-based access to other security risks present within projects across teams.

Turn functional tests into security tests

Integrating IAST into the pipeline allows you to run security tests parallel to functional or unit tests in preproduction environments, without additional testing workflows. Synopsys Seeker® IAST performs real-time security testing during the application's QA phase, observing communications and functions executed at runtime with the primary goal of identifying potential sensitive data leakage, insecure communications or data transmission protocols, and configurations that might be noncompliant with regulatory standards like PCI DSS or GDPR.

Seeker IAST provides an additional layer of security testing to complement SAST and SCA, which are not capable of detecting security issues at runtime or potential data leakage. As with Synopsys Code Sight, Coverity SAST, and Black Duck® SCA, Seeker IAST can leverage policies to uphold risk tolerance standards, trigger remediation events, and block promotion of insecure applications into production.

Centralize security risk insight and coordinate testing

With a variety of testing tools and workflows to integrate, it is imperative that security teams work toward a unified, centralized approach to application security testing. But legacy solutions from a panoply of vendors can complicate or delay this process. Working with security solution vendors that provide a portfolio with coverage for multiple testing tools is critical. The Synopsys Polaris Software Integrity Platform®, for example, integrates Synopsys security testing technologies (Coverity SAST, Black Duck SCA), centralizes test results and analysis, and provides a single-vendor way to establish DevSecOps strategies.

Security teams can leverage unified policies atop the Polaris platform to coordinate scans across DevOps pipelines, enforce security risk tolerance, establish automated security gates, and initiate remediation. This allows contributors from development, DevOps, and security teams to take the most appropriate action without wasted effort or deviating from preferred workflows.

Scale application security testing without increasing the burden

As you evolve your DevSecOps initiatives, it is important to plan for scalability and flexibility. This means preparing to support distributed development, security, and operations teams as well as the next iteration of your organization's technology stacks. Often, this means leveraging cloud-based, as-a-service solutions to optimize performance without the burden of maintaining infrastructure or up-front capital expenditure. The Polaris platform lets you consume security as a service, functioning as a centralized platform for application security testing with unified policies and risk insight for scaling AppSec.

The Polaris platform allows security teams to integrate security testing across DevOps pipelines and run the right tests at the right time. It supports testing technologies including Coverity SAST and Black Duck SCA, so security teams can configure policies to engage the appropriate scanning engine for the application being analyzed or pipeline event trigger. This approach makes your security testing strategy scalable as your organization grows and your technology portfolio evolves, so the investment you make in your security program now continues to benefit teams in the future.

Conclusion

DevOps is a foregone conclusion—organizations worldwide are accelerating development and evolving their technology portfolios for the next generation of software and cloud-native assets. Security must make a similar shift, but without sacrificing coverage or impeding DevOps pipelines. Complex development workflows, diverse testing tools, and distributed teams can challenge a concerted approach to DevSecOps. For security and DevOps teams everywhere, integrating testing technologies, defining contextual policies, and automating remediation workflows are emerging as the best mechanisms for balancing efficacy and efficiency. The success of these DevSecOps programs relies on centralizing high-quality risk data, ensuring complete testing coverage that's appropriate for the software passing through the pipeline, and establishing a strategy that can scale as your organization grows.

The Synopsys portfolio of application security testing solutions, developer-focused IDE integrations, and scalable software integrity platform establishes an end-to-end strategy for integrating security into DevOps workflows and CI/CD pipelines.

SYNOPSYS®

The Synopsys difference

Synopsys provides integrated solutions that transform the way you build and deliver software, accelerating innovation while addressing business risk. With Synopsys, your developers can secure code as fast as they write it. Your development and DevSecOps teams can automate testing within development pipelines without compromising velocity. And your security teams can proactively manage risk and focus remediation efforts on what matters most to your organization. Our unmatched expertise helps you plan and execute any security initiative. Only Synopsys offers everything you need to build trust in your software.

For more information, go to www.synopsys.com/software.