DevSecOps Strategy Guide:

Balancing Speed and Security

As modern software revolutionizes business operations and market competition, the pressure to satisfy aggressive time-to-market requirements has never been greater. The demand for effective security measures and reliable software has grown in tandem with the need to produce at speed. But the combination of rapid development and release cycles and increasingly sophisticated cyberattacks exposes a central challenge: how to develop and deliver quality software, at scale, without sacrificing security.

The key is incorporating application security testing (AST) into all stages of the software development life cycle (SDLC) and DevOps workflows. This sounds simple, but challenges remain, including managing the volume of overwhelming or redundant findings, unnecessary testing that can slow pipelines, subjective assessments of risk, and various levels of security awareness and security capability across teams.

It is imperative that contributors to security, development, and DevOps teams each have clearly defined roles for securing the digital assets they create, ingest, and push through their pipelines. Doing this in a way that scales as the organization evolves puts you in a position to thrive well into the future, but it requires a concerted approach to application security (AppSec). The traditional approach to AppSec, which "tacks on" testing, triage, and remediation to development, cannot keep up with the dangers posed by modern cybercriminals nor the workflow demands of modern DevOps.

Fulfilling the DevSecOps promise: Security at speed

There is a need for a new approach to AppSec—one that handles business risk without obstructing company growth, eliminates the mythical trade-off between speed and security, and fulfills the DevSecOps promise.

To realize this vision, organizations must create an efficient, effective DevSecOps strategy that incorporates three key ideas.

• Establishing application security across the SDLC and CI pipelines for continuous testing

• Maintaining development velocity without compromising on security

• Optimizing workflows to increase risk awareness and security capability across teams

This allows security mechanisms to be integrated into the SDLC at every level and provides the visibility and control that is essential for unified DevSecOps. It also aligns security and development teams to an objective assessment of risk and remediation priorities, and informs workflows that function smoothly across tools and roles.

So how do you get there? First, let's examine one of the biggest challenges to integrating security into DevOps: the diverse tools and systems used by each team.

Shifting security everywhere: Continuous testing at speed

From the applications on our smartphones to the systems that power organizations, industries, and governments, software has become an essential part of our daily lives. But software is not monolithic; it is a collection of components linked together by protocols and systems that enable the unified execution of operations and data transmission.

Your company's software, whether for internal use or provided externally to customers and partners, is composed of code you produce, software you borrow, and software you buy. The first is built by your developers to fit the distinct and individual needs of your company, and the second is made up of assets that your developers use to accelerate development. This second set of components consists of open source software and third-party libraries that can be used, updated, and distributed in accordance with licensing restrictions. Lastly, organizations purchase third-party licensed software binaries from external suppliers to incorporate into, or run alongside, their projects and systems.

Each of these software categories can be built using a variety of frameworks and technologies that convert code and components into operating programs. Containers for compartmentalizing elements of software functions, microservices for agile scaling, infrastructure-as-code (IaC) templates for defining cloud resource setups, API protocols, and other features are included.

Establishing software security becomes increasingly complicated with each new technology, since the mechanisms for testing and risk analysis vary depending on the technology. Consequently, maintaining high productivity and rapid release cycles becomes a more delicate balance with each additional test cycle appended to DevOps workflows. For most organizations, there is a baseline for AST that should form the foundation of your DevSecOps strategy.

• Detecting weak or insecure coding practices in proprietary code, and doing so as early as possible, at the developer desktop, to accelerate remediation and avoid pushing issues downstream

• Identifying known vulnerabilities in open source components, including transitive dependencies that may be unknowingly added into a project during a build

• Verifying the security of data and application functions at runtime, and detecting potentially malicious activity in running compiled assets

Each of these AST mechanisms identifies risks within distinct technologies and provides a critical level of insight for contributors to DevSecOps. But risk detection is only half of what is required to establish security—you also need effective risk remediation. Historically, risk detection was relegated to AppSec teams, and remediation to development teams.

In concerted DevSecOps programs though, these responsibilities interrelate, with AST becoming a continuous event that facilitates closed-loop communication between development, security, and DevOps teams as issues are prioritized for remediation and resolved through new or modified code. Doing this efficiently requires each testing technology and security mechanism to be incorporated into the SDLC, CI pipelines, and other DevOps workflows. How well this integration is done will impact each contributing team's performance and success metrics.

Eliminating friction: Integration and automation for unified DevSecOps

To fully achieve DevSecOps, each test needs to be integrated into established workflows to ensure that relevant testing is triggered while avoiding unnecessary testing that could delay software shipment. Effective DevSecOps also requires automated enforcement of risk-based policies based on objective standards of security. This provides some critical capabilities, including

- Integrated security gates that identify risks at or near their origin
- Security safeguards that identify risks as they enter main projects from extraneous or illegitimate workflows and pipelines
- Rapid and clear communication of prioritized risk analysis directly to developers for remediation
- Flexible and scalable DevSecOps that evolves with the organization

Automation is an essential part of the DevSecOps process. At its core, it allows AppSec, development, and DevOps teams to carry out their routine and requisite tasks with minimal manual intervention. For DevSecOps, automation must be based on deliberate, informed policies, risk tolerance thresholds, and purpose-built security gates that account both for security teams' and development teams' requirements (e.g., time-to-detection, time-to-remediation, code shipping deadlines). Properly defined AppSec policies are a fundamental mechanism to achieve your goals and make integrated, automated DevSecOps possible.

Because each team plays a unique role in defining policies in conjunction with security standards, DevSecOps initiatives will reflect a consistent, unified organizational vision even when a contributing party doesn't participate in a project or sprint. Policy violations warn stakeholders of an issue with a section of the project or workflow, and alert them that a predefined action is necessary.

Once policies and standards are established, you can integrate AppSec across your SDLC and CI pipelines. "Integration," in this sense, should be defined as connecting your AppSec tools to existing development, security, and DevOps systems and workflows. This includes

- Triggering AST scans based on pipeline activities, code changes, or new assets checked into source code management (SCM) and binary repositories, to accelerate risk detection
- Delivering prioritized risk analysis and fix guidance via issue management workflows and tools, to accelerate remediation
- Leveraging capabilities built into DevOps tools (e.g., GitHub Actions, GitLab Templates), to support automated testing, commenting, and fix pull requests
- Blocking the promotion of vulnerable assets or breaking pipeline builds that violate security policies, to preclude issues from manifesting downstream

By integrating these types of security and DevSecOps measures atop existing development and DevOps tools, processes, and software, you enable developers to work more efficiently, fix issues more quickly, avoid late-stage refactoring of code, and deliver more-secure, higher-quality software that minimizes your organization's risk exposure.

Of course, encouraging developers to embrace new tools and platforms can be challenging, particularly when those tools may be viewed as extraneous to developers' requisite daily tasks, as may be the situation with many AppSec testing tools. Leveraging integrations that allow security teams to seamlessly analyze software as part of existing development workflows, and delivering vulnerability information directly to developers within their IDEs, collaboration tools, and issue management workflows, greatly reduces the barrier to establishing security as part of DevOps and eliminates friction among crucial contributors.

Meeting developers where they are: Cultivating security capability

Developers are expected to work rapidly, and most of them have customized their IDEs and workflows to match their individual needs. Despite adding complexity to DevSecOps programs, these nuanced development environments and pipelines each share a common factor that can help establish more uniform security principles: the developers who engage with them.

As mentioned, integrating mechanisms for automated AST across the SDLC and CI pipelines helps establish security gates and a system of checks and balances to identify potential security issues as close to their origin as possible, regardless of their point of entry. You may, however, greatly enhance your organization's ability to secure its digital assets by cultivating security capabilities among developers, who often have widely varying skillsets for secure coding. The 2023 SANS report shows that nearly 21% of organizations feel that their security training for developers is inadequate, if they have a structured developer security training program at all. Yet only 31% view a developer security training program as a key success factor to a successful DevSecOps program.

Simply put, you can preclude the introduction of new issues and accelerate the remediation of existing issues by establishing a higher standard for security among developers, which they implement as part of their daily tasks. Achieving this as part of a DevSecOps initiative requires

- Access to clear, prioritized security risk insight as part of developer workflows, identifying the most urgent issues to fix and their location within source code and file systems
- Detailed remediation guidance to inform developers of the most efficient and effective ways to address detected issues, even when they lack the security knowledge or experience to do so unassisted
- Real-time risk awareness and fix recommendations as developers code within their preferred IDE, providing a "security spellcheck" experience to help developers write more-secure code before a merge or build
- Relevant developer security training and secure coding education that aligns to developers' projects, technologies, and greater business needs, fostering security capabilities that evolve with the business

> The combination of security awareness and security capability elevates an integrated DevSecOps initiative from a system of automated, triggered security gates to a concerted approach to agile risk detection and rapid resolution that becomes more efficient over time.

The combination of security awareness and security capability elevates an integrated DevSecOps initiative from a system of automated, triggered security gates to a concerted approach to agile risk detection and rapid resolution that becomes more efficient over time. A security capability based around best practices tailored to your organization, your technologies, and your regulatory standards helps replace a developer's subjective assessment of "secure" with a more objective assessment across projects and teams.

When issues are detected, developers should receive clear recommendations about how to fix such issues. This not only helps address the issue but establishes a standard that the developer can adhere to in the future. This should be reinforced by prescribed security training that applies to the technologies, languages, and frameworks each developer uses, maximizing the impact for the developer's learning efforts. This training should be consumed as part of existing development workflows, with easily consumed topical learning associated with security tests and directly accessible through integrations with issue management tools and IDEs, for example.

Lastly, it is important to activate these developer security capabilities as they code, with real-time notification of new issues they introduce within the IDE. You can achieve this with security testing that occurs locally on the developer terminal. Their code can be reviewed for coding weaknesses (e.g., cross-site scripting errors, no defined timeouts) and the inclusion of open source components or libraries with known vulnerabilities.

This local testing should not replace pipeline-based testing or centralized security testing by the AppSec team. It should, rather, serve as a mechanism to increase developers' awareness of issues that are often more-easily addressed earlier in development and that can delay or distract security teams during downstream application security testing. Consequently, the benefit is twofold: you accelerate time-to-remediation and reduce the burden on AppSec teams, all without requiring deviation from established development workflows.

Realizing DevSecOps: People, processes, and planning

DevSecOps relies on the successful collaboration and cooperation of a variety of roles within the organization. While AppSec teams may continue to take ownership of the security risk posture of an organization's software, their ability to implement security measures and address detected issues depends on development and DevOps teams. Many organizations stall when architecting a DevSecOps strategy because they fail to identify and incorporate the success criteria and performance metrics that matter to each contributing team.

For most, a cross-team discussion will yield a list of criteria and performance metrics similar to the following:

• Time-to-detection
• Time-to-remediation
• False positives in AppSec tests
• Cost of remediation

For modern DevSecOps though, this list doesn't provide actionable insight into the status of a security program and does not account for the inherent need for an integrated AppSec program to evolve with the organization. Concerns about speed can be addressed by by expanding the breadth and depth of your AppSec integrations across the SDLC, and by implementing policy-based automation False positive rates vary depending on the AST tool being used and the point at which such tests occur. And the cost of remediation costs can be influenced by proximity of the fix to the origin of the issue or the proficiency of the development teams tasked with the issue.

With all this in mind, it is important to identify each contributing team's success criteria and the DevSecOps mechanisms that can affect these criteria. Commonly, regardless of the critical factors valued by each team, the ability to assess the status and performance of a DevSecOps program by any given contributor is going to determine whether that contributor supports such a program and its supporting solutions.

The 2023 SANS survey reveals that more than 41% of respondents view organizational silos as a barrier to DevSecOps, with 36% identifying a lack of transparency as a pressing issue, despite 52% viewing cross-team communications as a critical success factor. Clearly, securing the support of contributing teams is going to be greatly influenced by your ability to allay any concerns that your DevSecOps initiative might be ineffective or inefficient.

You might consider an organizational approach to DevSecOps that reflects the distinct requirements of each team and supports a range of influential success metrics, including

• Smooth integration of security testing that not only avoids impeding development teams' deadlines, but fosters efficiency by avoiding late-stage rework due to failed security tests
• Developer visibility into new and existing issues that are prioritized for remediation and accessible within the IDE and issue management workflows
• Closed feedback loops between development and security teams for immediate notification of issue resolution
• Alerts of new issues that may impact existing software, so operations and DevOps teams can attest to the security posture of production assets
• Clear, manageable, and prioritized risk reporting that minimizes the number of false positives and reduces the vulnerability backlog
• It is possible to positively influence success criteria through a variety of actions, processes, and supporting technologies without fundamentally changing the requirements placed upon contributing teams. This is the hallmark of an effective, flexible DevSecOps program.

Creating solid foundations: Platform-based AST that evolves with your business

How can your organization remove complexity, reduce costs, and improve scalability without compromising security? Can this be done in ways that matter to developers, security personnel, and DevOps teams? Many organizations have been implementing cloud-based solutions for development toolchains due to their affordable, scalable, flexible, and easy-to-manage platforms and environments. Organizations want these benefits from their AST tools but historically, they have had to compromise on one or more of their basic demands. An intuitive platform might not be powerful enough to uncover security problems in complex applications, but a tool that is fast locally may not scale for enterprise. A hosted AST tool may perform well for static application security testing (SAST) but poorly for software composition analysis (SCA), or vice versa.

A SaaS-based AST platform can provide multiple analysis engines to help secure proprietary code and third-party software.

A SaaS-based AST platform can provide multiple analysis engines to help secure proprietary code and third-party software. With this approach, you can eliminate redundant tools or complicated implementations in lieu of a consolidated platform that is governed by centralized policies, integrates across CI pipelines, and provides prioritized risk insight across projects and tests. This is particularly valuable for organizations implementing a DevSecOps initiative, but that do not have the resources, need, or desire to manage on-premises implementations across all development, build, and testing environments.

Additionally, this ensures that your DevSecOps initiative is architected with flexibility and scalability from the start, without restriction or deviation due to the technological limitations of diverse AST tools. This as-a-service approach to platform-based AST combats the visibility and cross-team communication concerns identified by many as an impediment to DevSecOps. It also provides real-time insight into risk and remediation trends across projects and teams, making it easier to identify areas of concern and make necessary adjustments.

Enabling effective, integrated DevSecOps: Good business

The significance of application security extends far beyond safeguarding your software systems; it has evolved into a crucial business driver. Today, as a participant in a complex software supply chain, you must ensure the security of the software and components you ingest into your projects, while also recognizing that your customers and partners demand a reasonable attestation of security from your software. This cascade of security requirements means that your customers, partners, and end users are increasingly placing more value on your organization's risk posture and security controls, including

- Time-to-detection
- Time-to-remediation
- Accurate Software Bills of Materials with associated security attestation

If some of these look familiar, it is because there is an analogous requirement for DevSecOps initiatives to deliver on similar critical criteria for stakeholders. Many of the concerns held by your colleagues and contributing teams stem from the reality that they, too, are consumers of digital assets, just as your customers and partners are.

As your strategic vision and clearly defined DevSecOps program garners buy-in and contributions from others in your organization, so too does it become a potential business driver and competitive differentiator. A properly defined and implemented DevSecOps program clearly conveys

- Proactive security risk detection
- Responsive risk remediation
- Reliability and uptime of supported applications
- The security and inaccessibility of sensitive information that passes through the software and connected systems

By integrating DevSecOps practices, policy-based automation, and end-to-end visibility into risk and resolution for all contributing teams, you can quickly evolve AppSec from a cost center to business asset. In this landscape, AppSec directly impacts your organization's selling power, becoming a pivotal factor in differentiating you from your competitors. In essence, the ability to demonstrate robust security practices can either fuel or derail the revenue engine of a business, making it an imperative focus for any forward-thinking organization in our increasingly interconnected and security-conscious world.

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.