OpsMx

eBook

Four Effective Strategies for Optimizing Application Security with ASPM

In today’s digital age, the cybersecurity landscape is evolving at an unprecedented pace, presenting a complex array of challenges for businesses across all sectors. With the advent of sophisticated cyber threats and the rapid increase in attack vectors, traditional security models are being pushed to their limits.

Application Security Posture Management (ASPM) is a holistic approach to fortifying software applications throughout their lifecycle. This eBook outlines four effective strategies to optimize AppSec with ASPM:

1. Silencing the Noise: Effective Strategies to Tame Alert Overload
2. Empowering Developers: Smoothing the Transition with Shift Left
3. Simplifying Complexity: Unifying Siloed Data for Enhanced Security Visibility
4. Open Source Advantage: Customizing AppSec for Agility and Control

By implementing these strategies, organizations can enhance security, streamline workflows, and empower developers to build more secure and resilient applications

Limitations of Current AppSec Models

The limitations of current AppSec models are a sobering reality – many teams grapple with insufficient tools and processes to effectively close security gaps.

Traditional AppSec approaches are increasingly proving inadequate in scaling with these threats. Many security teams find their current tools and processes insufficient to keep pace with modern applications' rapid development and deployment cycles. This inadequacy stems from several core limitations:

- Inability to effectively manage the volume of alerts.
- High dependency on manual processes and interventions.
- The burden is placed on developers to incorporate security measures without impacting productivity.
- Tool sprawl and the challenges of managing multiple, often siloed, security solutions.

Key Statistics

- **75%** of security professionals have observed an increase in **cyberattacks over the past year. (CFO)**
- **Global cybercrime damage** costs are expected to grow by **15%** per year over the next two years, reaching **$10.5 trillion USD annually by 2025. (Forbes)**

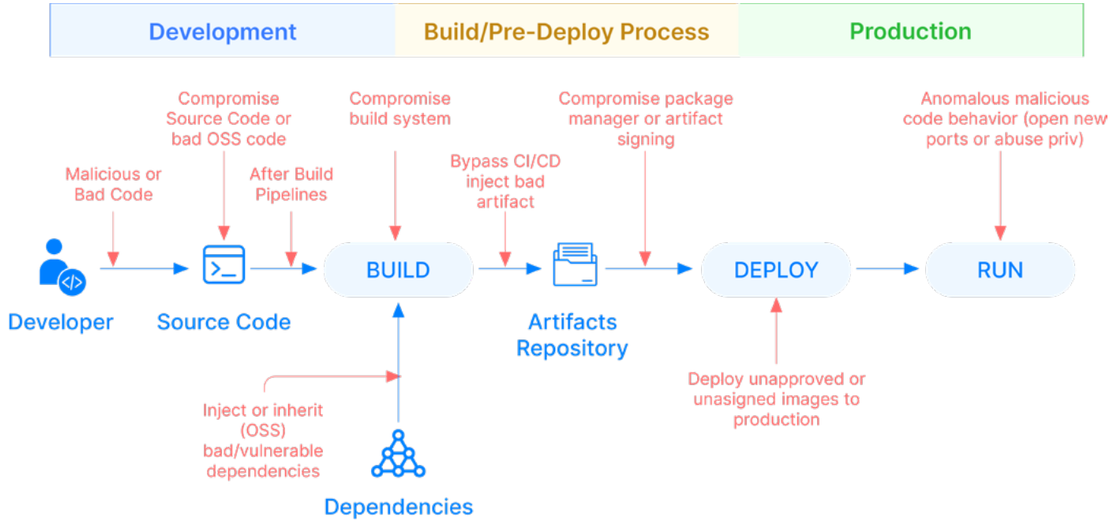


Figure 1: Real-World SDLC Security Threats

Application Security Posture Management is a comprehensive approach to enhancing the security of software applications throughout their lifecycle. It encompasses a variety of best practices and tools designed to assess, monitor, and improve the security posture of an organization's applications, crucial for integrating security into DevOps (DevSecOps). By integrating ASPM into CI/CD pipelines, organizations can detect vulnerabilities early, automate security controls, and maintain a unified view of application health, thereby managing risks effectively and building trust with users. This holistic approach is increasingly vital due to the complexity of modern software applications and the growing cybersecurity threats.

Key Components of a Successful ASPM Strategy

Given these challenges, there is a pressing need for a scalable and effective approach to Application Security Posture Management (ASPM). ASPM emerges as a comprehensive strategy designed to address the limitations of traditional AppSec models by focusing on automation, integration, and the strategic use of open-source tools. The key components of a successful ASPM strategy include:

- Proactive Alert Management: Strategically manages and prioritizes security alerts to actively prevent alert overload and reduce fatigue, efficiently addressing the high volume of alerts encountered in security operations.
- Automated Security Processes: Implement automation to minimize manual efforts in security checks, thereby alleviating the dependency on labor-intensive procedures and reducing the burden on developers.
- Holistic Security Integration Across SDLC: Seamlessly integrate security insights throughout the Software Development Life Cycle (SDLC) to ensure a unified view of application security, countering the issues arising from tool sprawl and siloed solutions.
- Strategic Open-Source Integration in ASPM: Optimize your security toolchain by integrating open-source tools within the ASPM framework. This approach curbs costs and enhances flexibility and transparency, enabling a focus on investing in ASPM platforms that provide a unified, value-driven view of security operations.

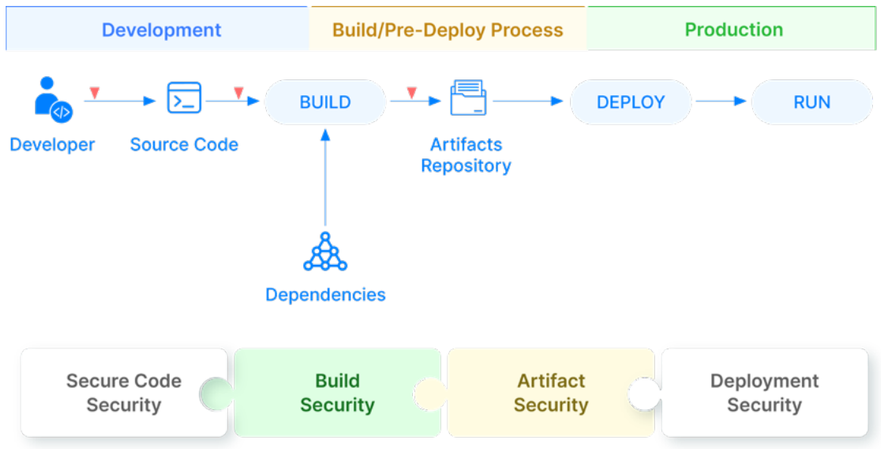


Figure 2: Securing the End-to-End SDLC

This eBook provides practical strategies for strengthening your application security posture using ASPM. Our goal is to empower AppSec teams with the insights and tools necessary to confidently overcome common ASPM challenges, resulting in more secure and resilient applications.

01

Silencing the Noise: Effective Strategies to Tame Alert Overload

Alert overload and fatigue pose significant challenges for AppSec teams. A constant flood of miscategorized security alerts, ranging from trivial to critical, overwhelms analysts and impedes their ability to discern actual threats swiftly. This barrage, compounded by information overload, significantly complicates the extraction of actionable insights, further straining resources and efficiency.

> **Alert fatigue overwhelms AppSec teams, hindering quick threat identification amidst constant noise and information overload.**

The Sheer Volume

A 2023 study by the Ponemon Institute found that the average enterprise processes over **17,000** malware alerts in a typical week. This constant influx hinders the ability of teams to maintain focus and effectively prioritize potential threats.

The Issue of Alert Overload

The ever-growing number of security tools in the software delivery pipeline contributes significantly to the volume of alerts generated. While each tool serves a specific purpose, not every alert signals critical risk. This constant stream of notifications challenges AppSec teams, potentially burying severe threats within a sea of noise.

The False Positive Dilemma

Teams are overwhelmed by a barrage of security alerts, many of which are false positives that misclassify benign activities as threats. These erroneous alerts flood systems, obscuring critical warnings and hindering security operations. False positives are a significant drain on resources, forcing security teams to spend valuable time investigating them instead of real threats. This misallocation can delay responses to actual vulnerabilities, potentially leading to undetected breaches, data loss, financial damage, and erosion of customer trust.

> **The crux of navigating this overload is refining the 'Signal to Noise' ratio, ensuring that critical alerts stand out.**

False positives persist due to aggressive detection settings, outdated threat definitions, and a lack of contextual awareness by security tools. Each false alarm not only wastes time but also impacts the morale and effectiveness of the security team, leading to alert fatigue and increased risk of oversight.

The Business Impact

The consequences of alert fatigue cannot be overlooked. Overwhelmed teams increase the risk of missing critical breaches, leading to potential data loss and damage to reputation. Furthermore, burnout and resource strain contribute to higher turnover rates within cybersecurity teams – a serious concern given the skills shortage in the field.

Effective Alert Management Strategies

A flexible and customizable alerting framework is central to overcoming alert overload. This adaptability ensures that alerts are relevant and actionable, aligning with the organization's unique needs.

> **Best Practice:** Define what constitutes a critical alert based on the specific risk profile and operational context of your software delivery processes.

Prioritization Techniques

Given the sheer volume of alerts, prioritization becomes indispensable. Techniques that assess the severity, exploitability, and impact of identified vulnerabilities can help AppSec teams allocate their efforts where they are most needed.

- **Risk-Based Approach:** Categorize alerts based on severity, exploitability, and impact to ensure high-priority issues receive immediate attention.
- **Prioritization Based on Business Context:** One of ASPM's key capabilities is its ability to prioritize alerts based on the broader application and business context, not just severity. This means that ASPM tools should help organizations decide which issues need immediate attention and which can be deferred based on their potential impact on the business. This approach helps reduce the alert fatigue that security teams often face.

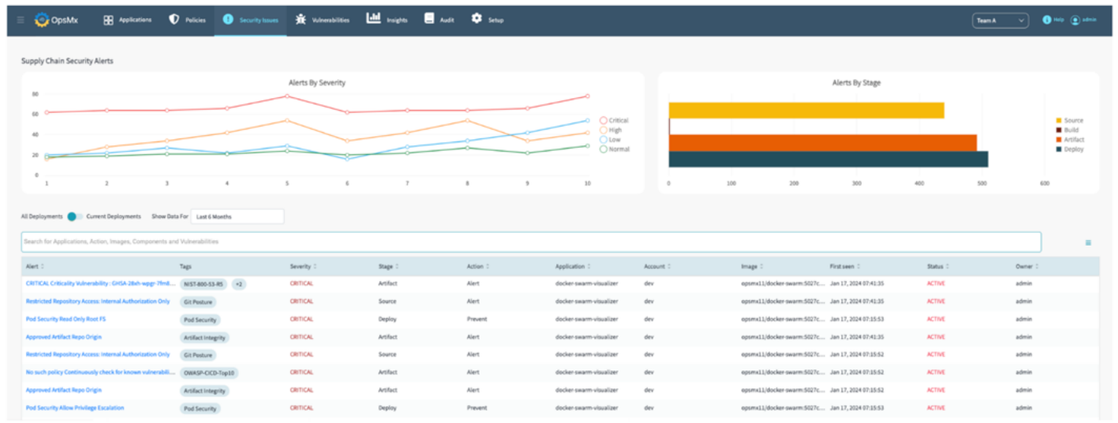


Figure 3: Security Alerts and Trends

Leveraging Automation for Critical Alerts

In the dynamic environment of application security, automation can swiftly respond to critical alerts to mitigate risks before they escalate. By implementing intelligent automation workflows, organizations can focus on high-impact tasks while routine alerts are managed systemically.

- **Automated Triage and Response:** Configure your systems to categorize alerts by severity automatically. High severity alerts can trigger immediate investigations or remediation actions, such as patching known vulnerabilities or isolating affected systems.
- **Integration and Orchestration:** Embedding automated security within the development pipeline ensures that security checks are a natural part of the development process, occurring in real-time with minimal disruption.
- **Use Case in Automation:** Consider an automated system that, upon detection of a critical vulnerability, not only alerts the security team but also cross-references the vulnerability with deployed patches, reducing false positives and focusing efforts on unresolved issues.

> **Embracing automation across our toolchain transforms our security posture from reactive to proactive, significantly reducing the need for manual intervention and allowing us to focus on strategic security initiatives.**
>
> **Bob Boule**, VP Of Products, OpsMx

User-driven Alert Management

User-driven alert management acknowledges the expertise of individual team members, giving them control over the security alerts they receive. This approach ensures that alerts are received by the right people and actioned promptly and efficiently.

Role-based Alert Routing: Implement a system where alerts are routed according to the role and expertise of team members, ensuring that the right alerts reach the right experts at the right time.
Customizable Alert Subscriptions: Enable team members to subscribe to specific types of alerts relevant to their area of responsibility. This personalization reduces noise and focuses attention on areas where an individual can have the most impact.
Example Use Case for Pub/Sub Model: A developer working on payment systems can subscribe to alerts related to financial data security, receiving immediate updates on new threats or vulnerabilities in payment-related code libraries.

By refining these approaches, organizations can create an alert management ecosystem that is both responsive and responsible, catering to the strengths and needs of the security team and ultimately fostering a culture of security-mindedness across the entire organization.

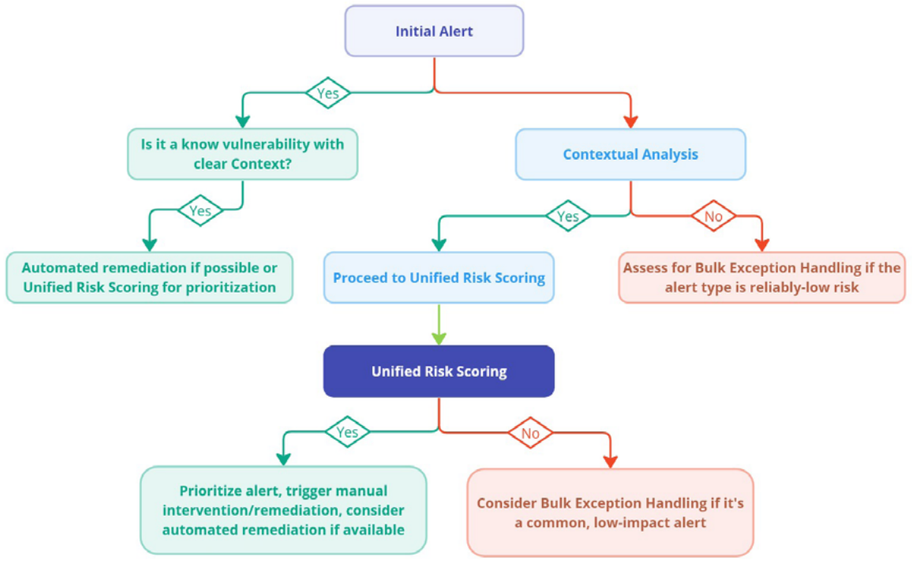


Figure 4: Alert management flowchart

Example of Strategies to Minimize Alert Overload

To improve alert management further, several strategic approaches are mapped to the flowchart, with strategies detailed below.

1. **Automated Remediation and Risk Scoring:** When a known vulnerability with clear context is identified, automated remediation is initiated, or the issue is escalated based on a unified risk score.
2. **Contextual Analysis for Clarity:** If the alert lacks context, ASPM tools perform a contextual analysis. Alerts that can't be clarified may be bulk handled if deemed low-risk.
3. **Actioning on Unified Risk Scoring:** For alerts that pass contextual analysis, a unified risk score determines the priority and the necessary response—whether immediate manual action or additional automated remediation.
4. **Ongoing Monitoring for Assurance:** Continuous monitoring across all stages ensures that fixes are effective and new vulnerabilities are identified promptly, maintaining a robust security stance.

Navigating the challenge of alert overload in ASPM necessitates a multifaceted approach that balances the customization of alert management with the strategic use of automation and prioritization techniques. By refining the "Signal to Noise" ratio and overcoming tool and data silos, AppSec teams can ensure that critical alerts receive the attention they require, enhancing the overall security of the software delivery process.

Empowering Developers: Smoothing the Transition with Shift Left

The adoption of the Shift Left philosophy in DevSecOps practices, intended to embed security early in the software development lifecycle (SDLC), has undeniably brought security into the forefront of development processes. However, this strategic shift has also introduced significant challenges for development teams, compounding their pressures in an already demanding environment.

Core Challenges Exacerbated by Shift Left:

- **Manual Intervention:** Despite advances in automation, manual security processes remain prevalent, consuming valuable development time and resources. This slows down the SDLC and diverts developers from their primary focus—building and refining software.
- **Complex Security Tooling:** The integration of multiple security tools, each with its own learning curve and maintenance requirements, adds complexity. Developers must navigate these tools while meeting their project deadlines, often without adequate training in these specialized security applications.
- **Insufficient Security Expertise:** With the Shift Left approach, developers are expected to take on security responsibilities traditionally handled by security professionals. This expectation overlooks the specialized nature of security work and places an unrealistic demand on developers who may not have the necessary background in security.

- **Slower Development Cycles:** Increased workloads and context-switching can lead to delays in feature releases.
- **Compromised Innovation:** If developers are constantly firefighting security issues, less time is available for creative work and innovation.
- **Burnout and Frustation:** Overburdened developers may become demoralized or experience burnout. This can hurt morale and productivity.
- **Security Debt:** If teams postpone or insufficiently address security issues, a backlog of security risks (security debt) can build up, creating larger future problems.

To truly achieve the goals of Shift Left in DevSecOps, it's imperative to effectively align the development teams with these new security practices. Simply increasing their workload is not a sustainable strategy. AppSec teams must deliver solutions that not only bolster security but also enhance and accelerate the work of development teams. The success of ASPM hinges on its acceptance by developers, who must see it as a facilitator, not a hindrance, making their tasks easier and more efficient.

Enabling Developers with ASPM Insights

A key advantage of Application Security Posture Management (ASPM) is its ability to swiftly identify vulnerabilities, such as Common Vulnerabilities and Exposures (CVEs), within the development process and provide context within the application. This allows developers to address these issues promptly, preventing them from escalating into more significant problems. By integrating ASPM solutions into the development lifecycle, organizations can reduce the burden on developers, enhancing both productivity and security.

Streamlined Workflow: Real-Time CVE Identification and Remediation

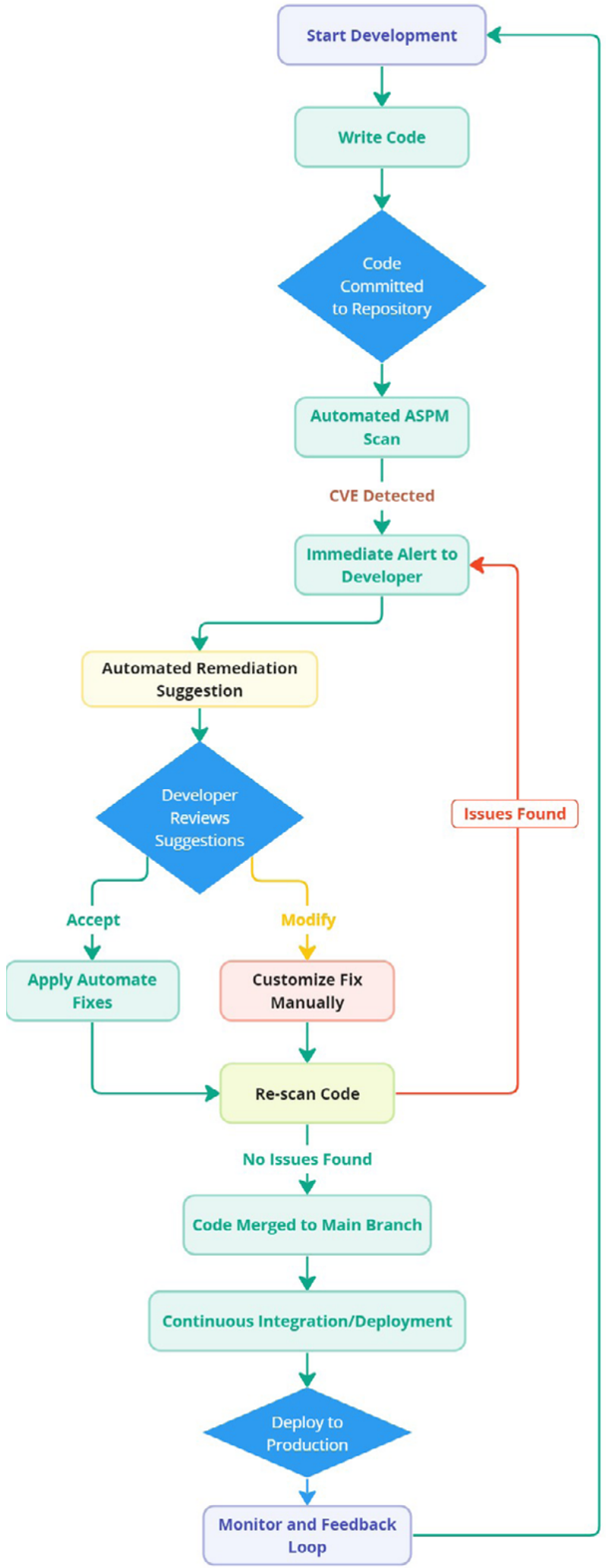
The integration of ASPM solutions into development environments significantly streamlines the process of identifying and resolving security issues. Here's a detailed breakdown of how this integration benefits developers:

- **Integration into Development Environments:** ASPM tools are seamlessly incorporated into the environments and IDEs developers already use, minimizing disruption and learning curves.
- **Real-Time Scanning:** While developers write code, ASPM tools continuously scan for potential vulnerabilities, providing immediate feedback without waiting for scheduled scans.
- **Immediate Feedback:** When a CVE is detected, the tool alerts the developer with contextual information about the vulnerability and potential fixes directly within their workflow.
- **Guided Remediation:** Developers receive concise, step-by-step instructions on remedying the issue, integrated into their current projects, promoting quick and efficient resolution.
- **Continuous Monitoring:** Following a fix, ASPM tools persist in monitoring the code to verify the resolution and continue checking for new vulnerabilities.

This workflow exemplifies how ASPM can expedite the detection and resolution of security issues, allowing developers to maintain their development pace while enhancing security.

Figure 5:

Developer workflow



Explanation of Each Step in the Flowchart:

1. **Start Development:** The process begins when development starts.
2. **Write Code:** Developers write code in their local environment.
3. **Code Committed to Repository:** Code is committed to a version control repository.
4. **Automated Security Scan:** Automatically scan the committed code for security vulnerabilities using security tools.
5. **Immediate Alert to Developer:** If vulnerabilities (such as CVEs) are detected, an immediate alert with details is sent to the developer.
6. **Automated Remediation Suggestion:** ASPM tools suggest fixes or patches for the detected vulnerabilities.
7. **Developer Reviews Suggestions:** The developer reviews the suggested fixes.
8. **Accept:** If the suggestion is adequate, the developer applies it.
9. **Modify:** If the fix needs adjustment, the developer customizes it manually.
10. **Apply Automatic Fixes/Customize Fix Manually:** Depending on the action taken, the developer applies the automated fixes or manually adjusts them as necessary.
11. **Re-scan Code:** The modified code is re-scanned to ensure no new issues have arisen and all previous issues are resolved.
12. **Code Merged to Main Branch:** Once the code passes the security checks, it is merged into the main branch.
13. **Continuous Integration/Deployment:** The code undergoes continuous integration and deployment processes.
14. **Deploy to Production:** Successfully integrated and tested code is deployed to the production environment.
15. **Monitor and Feedback Loop:** Post-deployment monitoring occurs, with feedback loops in place to ensure any new issues are promptly addressed, leading back to further code development.

This flowchart demonstrates how integrating ASPM into the SDLC can significantly streamline the security aspect of software development, enhancing both the development process's efficiency and security posture.

Effective ASPM Strategies for Reducing Developer Burden

Here is a table outlining effective ASPM strategies designed to minimize developer burden and maximize productivity by integrating security more deeply and seamlessly into the development process:

Strategy	Description	Impact On Developers
Automated Real-Time Scanning	Incorporate ASPM tools that perform continuous security checks directly within the developers' IDEs and commit pipelines.	Minimizes disruptions by providing instant feedback and security insights without leaving the development environment.
Developer-Centric Security Guidance	Deliver tailored security advisories and fix suggestions directly within the tools developers use daily, such as through IDE plugins or during code reviews.	Enhances developer efficiency by providing clear, actionable security instructions alongside regular coding tasks.
Proactive Security Integration	Embed security features and checks into the standard development tools and platforms to ensure security is a natural part of the development lifecycle.	Reduces the need for separate security steps, making security a seamless aspect of daily activities.
Collaborative Security Practices	Facilitate a cooperative environment where security and development teams collaborate closely through tools and regular interactions to enhance security without imposing additional burdens.	Promotes understanding and swift resolution of security issues, making security a collective responsibility and benefiting from diverse expertise.
Enhanced Training and Support	Provide ongoing training and support tailored to the development environment and languages used, focusing on security practices relevant to the developers' daily tasks.	Builds competence and confidence in handling security issues, reducing the perceived burden and fostering a proactive security posture.

Transforming Developer Experience with ASPM

Through strategic integration of ASPM, organizations can significantly alleviate the burden on developers, transforming security from a disruptive requirement into a beneficial enhancement of their workflows.

These strategies ensure that developers are equipped to handle security issues efficiently and are empowered to contribute to the security posture proactively.

The goal is to make ASPM tools indispensable to developers by making their jobs easier, not harder—turning ASPM into a solution that developers want and demand because it unequivocally enhances their productivity and reduces their workload.

Simplifying Complexity: Unifying Siloed Data for Enhanced Security Visibility

As the software development lifecycle (SDLC) rapidly evolves, developers face a proliferation of security tools, leading to siloed data and fragmented visibility. This sprawl complicates effective security management. By strategically applying Application Security Posture Management (ASPM), it's possible to integrate these varied tools, ensuring cohesive data context and streamlined workflows throughout the entire development pipeline.

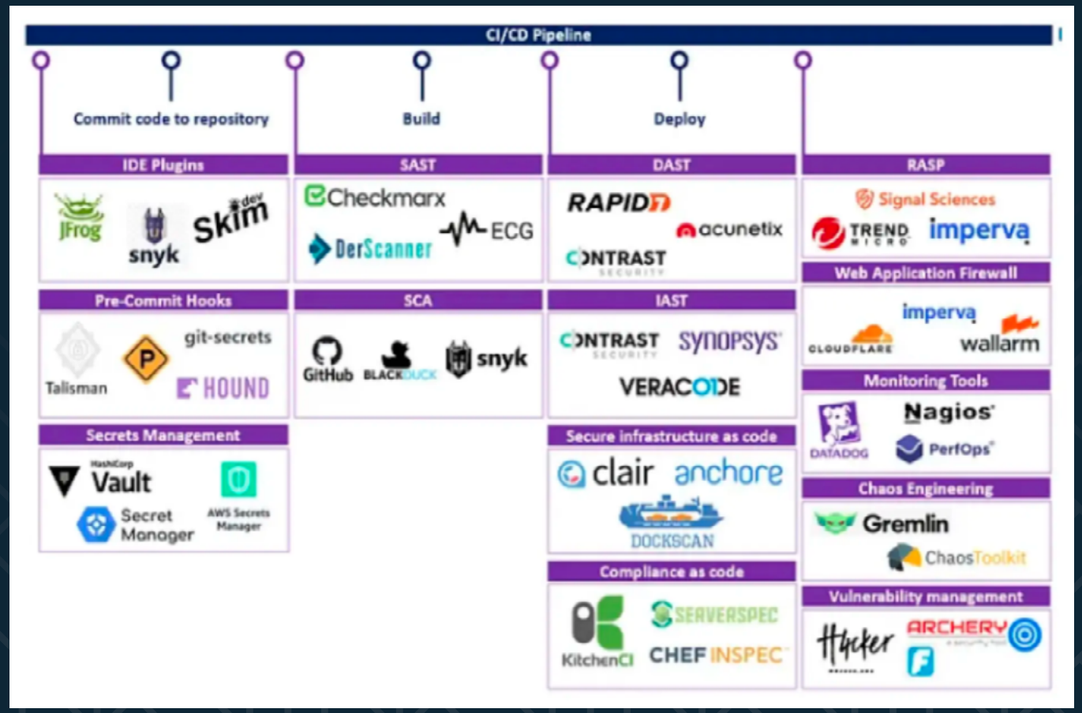


Figure 6: SDLC gets complicated

Navigating Tool Sprawl

In the modern software development environment, tool sprawl is an escalating problem. Organizations find themselves saddled with an array of security solutions—each promising to plug different security gaps. This proliferation leads to a complex security ecosystem where critical data gets locked in tool-specific silos, hampering both visibility and efficiency.

Challenge of Complexity: Developers often struggle to navigate through a maze of security tools, each with its own interface, requirements, and data outputs. The learning curve is steep, and integration points are numerous, leading to increased setup time and potential for misconfiguration.
Reduced Efficacy: When security data is scattered across multiple platforms, it's challenging to get a holistic view of the security posture. This fragmentation can lead to missed vulnerabilities, as insights from one tool may not correlate with data from another.
Strategic Consolidation: Addressing tool sprawl necessitates a thoughtful strategy of consolidation. By selecting tools that integrate well with each other and support the broader goals of ASPM, organizations can simplify their security toolchain without sacrificing coverage.

Achieving Unified Visibility

Unified visibility is the linchpin of an effective ASPM strategy. It transcends the limitations of tool sprawl by integrating data and toolsets to offer a comprehensive security overview. This integration empowers organizations to make more informed decisions and respond to threats with agility.

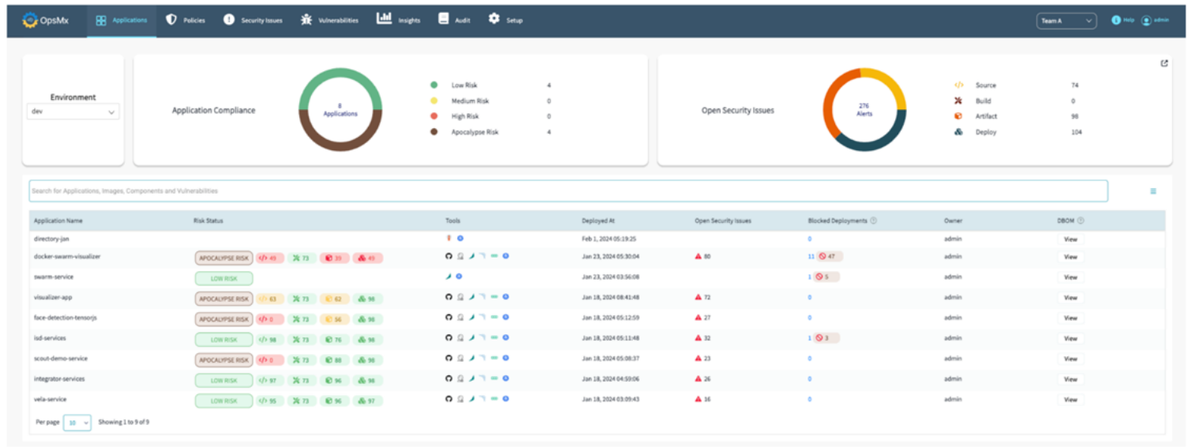


Figure 7: Example of unified application visibility

Data Integration: Centralizing data from various tools into a single repository is a vital step toward unified visibility. It enables real-time analysis and contextual understanding of security data, which is crucial for proactive threat management.
Holistic Context: ASPM provides the context necessary to understand the significance of security data. By combining insights from various stages of the SDLC, ASPM platforms can pinpoint where vulnerabilities are likely to impact the application most severely.
Streamlined Workflows: A unified view allows for the creation of streamlined workflows. Tasks that once required accessing multiple systems can now be managed from a central console, boosting productivity and reducing response times.

Benefits of Unified Visibility

Faster Response Times
When all security data is visible in one place, the time to detect and respond to threats is significantly reduced.

Improved Threat Detection
Integrated data leads to better threat detection, as patterns and anomalies that were previously obscured by tool silos become apparent.

Enhanced Compliance Posture
Unified visibility simplifies compliance by providing clear, consolidated reporting on security measures and incident responses.

</>

Developer Empowerment
With accessible, unified security data, developers can make more informed decisions about code and architecture, embedding security into the application's fabric.

Leveraging ASPM to Drive Value

The true value driver of ASPM lies in its ability to simplify and strengthen the security toolchain. By serving as the integrative force behind the myriad of tools, ASPM enables organizations to overcome the limitations of tool sprawl, delivering actionable security insights across the SDLC.

Unified Risk Scoring: Incorporate risk scoring methodologies that aggregate findings from multiple tools to prioritize threats effectively.
API-Driven Integration: Utilize APIs to weave disparate tools into a cohesive fabric, ensuring data flows freely and securely between them.
Orchestrated Remediation: Create automated remediation pathways that leverage insights from the unified view, allowing for rapid, coordinated action against threats.

Best Practices for ASPM Implementation

Unified visibility is the linchpin of an effective ASPM strategy. It transcends the limitations of tool sprawl by integrating data and toolsets to offer a comprehensive security overview. This integration empowers organizations to make more informed decisions and respond to threats with agility.

Selection Criteria: Choose tools not just for their standalone capabilities but also for their ability to integrate into the ASPM framework.
Vendor Collaboration: Engage with vendors to ensure that their tools are designed with integration in mind, facilitating a more seamless ASPM experience.
Training and Enablement: Equip teams with the knowledge and skills to utilize ASPM platforms effectively, translating unified visibility into operational security improvements.

Tool sprawl within the security toolchain is a complex challenge but one that can be addressed with the strategic application of ASPM. By consolidating tools and achieving unified visibility, ASPM becomes a force multiplier in the SDLC, enabling quicker responses, clearer insights, and a more secure end product.

Open Source Advantage: Customizing AppSec for Agility and Control

Within the dynamic context of application security (AppSec), the choice between open-source and proprietary software has far-reaching technical implications. As attack vectors multiply and security requirements intensify, open-source software (OSS) provides compelling benefits worth serious consideration by security practitioners.

The inherent transparency of OSS is a major advantage. Access to the source code enables security teams to conduct in-depth audits, uncover potential vulnerabilities, and tailor the software to their specific security needs. This open structure encourages a collaborative environment where security patches and improvements can be developed and integrated rapidly.

Advantages of Open-Source for AppSec

- **Security and Open Source:** Open source software invites transparent peer review, often resulting in more rapid identification and remediation of security vulnerabilities.
- **Open and Flexible:** The malleability of OSS allows for tailor-made security solutions, ensuring that the software can evolve alongside emerging threats.
- **No Vendor Lock-In:** Freedom from vendor lock-in enhances an organization's agility in adapting to new security challenges without contractual limitations.
- **Lower Cost:** The cost-effectiveness of open source tools frees up financial resources, allowing for investment in other crucial security areas.

Moreover, the absence of licensing fees and vendor lock-in associated with OSS can free up significant organization resources, allowing for greater investment in proactive security measures and ongoing innovation.

Navigating Tool Sprawl

Feature	Open Source Software	Proprietary Software
Cost	Generally lower cost with no licensing fees.	Higher costs due to licensing fees and potential additional charges for updates.
Flexibility	High; source code access allows custom modifications.	Low; dependent on vendor for updates and customizations.
Innovation Speed	Fast; benefits from community contributions.	Slower; limited to vendor's timelines and innovation strategies.
Vendor Lock-In	None; freedom to switch tools and adapt solutions.	High; switching costs and dependency on vendor's roadmap.
Security	Transparent; community scrutiny helps identify and fix vulnerabilities quickly.	Opaque; relies on vendor for security updates and vulnerability management.

Table 2: Comparative analysis: OSS vs Proprietary Software

Security and Open Source: Transparent Vulnerability Management

Open source software (OSS) is predicated on the principle of transparency. It offers an open platform where security vulnerabilities can be identified and addressed collectively. Unlike proprietary software, the source code in OSS is available for anyone to review, which means a larger pool of developers can scrutinize the code for potential security flaws. This collective vigilance often leads to more secure software.

Open and Flexible: Custom-Fitted Security

OSS's inherent flexibility is a tremendous asset for security teams. It allows organizations to modify and customize the code to fit their specific security needs. As threats evolve, security measures can be adapted without waiting for a vendor's update cycle.

No Vendor Lock-in: Agile Security Posture

The absence of vendor lock-in with OSS gives organizations the agility to switch between different tools or adapt them as required without being constrained by licensing agreements. This ensures that their security infrastructure can quickly adapt to new challenges.

Lower Cost: Maximizing Security Budget Efficiency

Open source tools can significantly reduce costs, providing a financial advantage that enables organizations to reallocate their budgets toward other critical security areas, such as employee training or advanced threat detection systems.

Job to be done	Average Cost Of Commercial Alternatives
Static Code Analysis	$10,000 - $50,000
Secret Scanning	$5,000 - $20,000
Binary / Image Scanning	$2,000 - $50,000
Artifact Mangement With Continuous Security Scanning	$2,000 - $50,000
Environment and IaC Security	$5,000 - $25,000
Build and Deployment	$50,000 - $100,000

Figure 8: Potential cost savings for a medium size company

Common Open-Source AppSec Tools

SAST tools analyzes source code for vulnerabilities, detects flaws early to enhance security, and reduces remediation costs in the SDLC.

GitLab SAST

SonarQube

Bandit

ESLint

Brakeman

Software Composition Analysis (SCA) tools identify vulnerabilities and license issues in open-source dependencies, helping secure applications by scanning libraries, package manifests, and third-party components.

Snyk

Trivy

OWASP Depen-dency-Check

OWASP Depen-dency-Check

Dynamic Application Security Testing (DAST) tools analyze running applications for vulnerabilities by simulating attacks, identifying security flaws like SQL injections and cross-site scripting in real-time.

ZAProxy

Arachni

Nikto

sqlmap

Secrets management tools securely store and manage sensitive data like API keys, passwords, and certificates, ensuring encrypted access and minimizing unauthorized exposure in applications.

HashiCorp Vault

Conjur

Bitwarden

Keywhiz

Vulnerability Management involves identifying, assessing, prioritizing, and remediating security vulnerabilities in systems and applications to reduce exposure and strengthen an organization’s cybersecurity posture.

OpenVAS

Clair

Nessus Essentials

Open-source software offers a potent combination of cost efficiency, flexibility, and collaborative security enhancement. For organizations committed to robust AppSec practices, integrating open-source tools into their ASPM strategy can lead to significant improvements in security posture without the constraints associated with proprietary solutions.

Compliance as Code automates security and regulatory compliance checks within the development process, ensuring consistent standards enforcement across IT infrastructure.

Ansible

InSpec

OpenScap

Infrastructure security involves safeguarding IT systems and networks by implementing measures like firewalls, access controls, and encryption to protect against unauthorized access and attacks.

Lynis

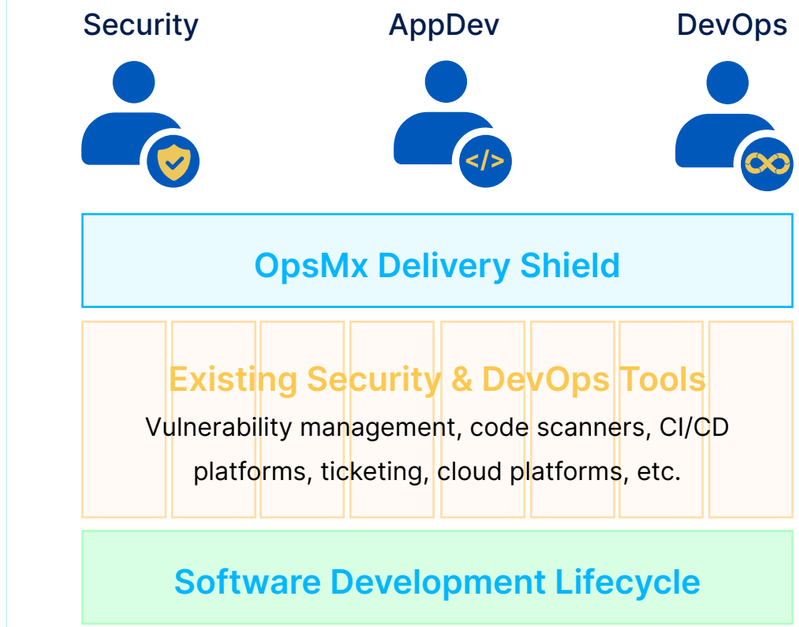
OSSEC

How OpsMx Can Elevate Your AppSec Strategy

OpsMx excels in securing your SLDC from developer to deployment, leveraging an Open Software Delivery architecture and AI/ML-powered DevSecOps solution. We aim to help organizations ship better software faster, with heightened security and compliance.

OpsMx Secure Software Delivery Overview

OpsMx Delivery Shield enhances your application lifecycle with continuous security posture management, providing global visibility and strict policy enforcement. The challenges in today's AppSec environment include tool fragmentation, disjointed data, and the burden of 'shift left' on developers. OpsMx addresses these by facilitating faster, more secure application releases, automated compliance, and reduced overall costs.



Modern development and delivery demands seamless integration of security practices to avoid bottlenecks. OpsMx tackles this challenge by embedding itself within the SDLC workflows. This fosters "shift left" practices, empowering developers with real-time security feedback. By providing actionable security recommendations, OpsMx enhances developer productivity and visibility. This allows teams to prioritize rapid development and AppSec security standards adherence.

OpsMx captures every step in the process, enabling Automated Compliance reporting and evaluation, including the Delivery Bill Of Materials.

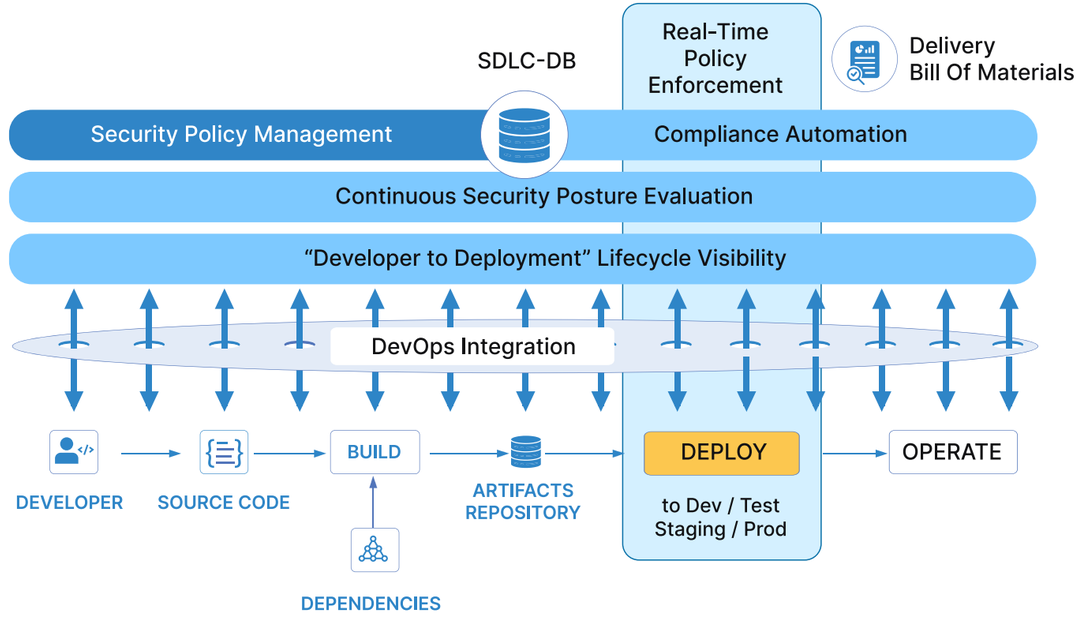


Figure 10: OpsMx Application Security Architecture

OpsMx Delivery Shield takes a comprehensive, developer-to-deployment approach to ensure application security and compliance, including:

- **Application Lifecycle Visibility.** Many organizations already have the security data they need spread across their existing tools and processes. OpsMx consolidates and analyzes that data in one place.

- **Security Posture Evaluation.** A moment in time security checks are not enough. OpsMx continuously monitors security risks in application releases across dev, test, staging, and production environments.

- **Policy Enforcement.** Control the release process with automated approvals and release verification, as well as block high-risk releases.

- **“Shift Left” Developer Productivity.** Give developers more time to code with actionable guidance on addressing security gaps.

- **Incident Response.** New vulnerabilities can be announced at any time. With OpsMx, you can find them faster and fix them sooner

- **Security Effectiveness and Compliance.** How well is the organization following its security policies and best practices? Replace manual data collection and compliance reviews with on-demand reporting.

Our team is ready to provide a demo and assist you in implementing effective application security posture strategies. Embrace a proactive approach to application security posture management and safeguard your software supply chain. Contact us today!

About OpsMx

OpsMx simplifies and intelligently automates secure software delivery, enabling hundreds of thousands of developers at Google, Cisco, Western Union, and other leading global enterprises to ship better software faster. OpsMx is the first platform designed to securely deploy applications in container, virtual machine, and multi-cloud environments. The company’s 120 employees serve customers from Silicon Valley, Hyderabad, and Bengaluru offices, with funding from Dell Technologies Capital and Foundation Capital. For more information, visit opsmx.com

OpsMx