Manage AppSec Risk at Enterprise Scale

More software means more risk to manage

The software industry continues to show unprecedented growth and transformation. With digitalization, cloud adoption, and the rise of AI, software has become the backbone of businesses across all sectors. However, the rapid proliferation of software has introduced new challenges for development and application security (AppSec) teams.

The acceleration of software development both for internal and external purposes means that organizations must secure a growing number of applications. Over time, this has resulted in a proliferation of security tools. As a result, teams are managing large, complex, heterogenous application security testing (AST) environments where hundreds of applications are being tested by dozens of tools, often across multiple teams. They are faced with more tools, running more tests, and delivering more results, which can result in a lot of noise. The struggle to manage, support, prioritize, and report on overall software application risk is very real.

And as the amount of software has increased, so too has the attack surface malicious actors target. Further, the addition of developments like AI-generated code creates an increasingly complex software supply chain. More code leads to more risk end-to-end. When this increased risk is paired with less visibility into where that code comes from, and the security risks that poses, there is even more complexity for AppSec teams to manage.

On top of the criticality of software and the increased attack surface, there is also added regulatory pressure. Some regulations have been in place for quite some time, such as PCI and HIPPA, but teams are also sorting through the requirements of the executive order on software supply chain security, secure software development frameworks, and DHS-CISA self-attestation. This all amounts to increased scrutiny and monumental reputational, financial, and legal risks for teams that are struggling to maintain and attest to the security of their applications.

Many of these developments are not new. The growth of software has been happening for many years, and as the landscape of security tools has expanded alongside, the task of managing all this at the scale an enterprise demands has grown increasingly complex. The budgetary and resource pressure put on teams in light of the current economic climate adds real urgency. Many teams are being asked to secure more software, assembled across an increasingly complex supply chain, with increased regulatory pressure, and no increases to budget or headcount.

Simplification and consolidation are key to addressing these issues. They are increasingly being looked at as ways to decrease overall total cost of ownership (TCO), improve overall risk posture, and provide teams with a rapid time to risk insight.

“Shift left” has evolved into “shift everywhere”

As software development has evolved over the years from waterfall to agile, so has security. Security used to be an afterthought, with some activities performed in the QA phase right before deploying an app into production. In the early 2000s, that changed. The idea of “shift left” was introduced as a means of increasing security without slowing down velocity, but this process was hampered by the lack of in-depth testing. In those early days of shift left, teams only really had access to static analysis for testing during the coding phase of the software development life cycle (SDLC).

In 2020, the “Building Security in Maturity Model” (BSIMM) report first documented the initiative to “shift everywhere.” Shift everywhere engaged the software security program (SSP) to perform security testing as soon as an artifact was available, no matter where that availability fell in the SDLC. As digital transformation has demanded software innovation across all sectors, shift everywhere has become the dominant model that security experts now advocate for when it comes to security activities.

As a result, organizations are testing on a continuous basis to increase the efficiency and effectiveness of security testing. Application security activities have increased across the SDLC correspondingly, and development teams are now tasked with onboarding, learning, and running a whole suite of security tools on top of their regular development workflows.

Because shift everywhere means AST everywhere, AST has to collectively function as a telemetry generator, a single framework that’s part of the engineering plumbing. AST results must be curated and presented to engineering teams when, and only when, prioritization indicates that issues must be addressed ***now***. That telemetry must also extend beyond just AST. The entire security function, from automated AST to procedural and adherence issues, must operate from a central point to ensure consistency and provide the necessary regulatory controls. Relying on spreadsheets, phone calls, emails, and expensive group meetings rather than automation for security gates is one of the many things that add cost and decrease risk insight in an organization.

The challenge of security at velocity

How do organizations balance security and velocity when team goals for product security and development velocity conflict? The goal of developers is speed—they are usually assessed on how quickly they build features and push them to production. Historically security has not been a part of their job, and even as security shifts everywhere, a perennial challenge for security teams is that development teams often seek to define noncritical defects as technical debt that can be settled later. Meanwhile, if the security team is automating policy and setting gates to test artifacts as soon as they’re available, every development team is experiencing more tests, on more artifacts, at more stages in the development cycle. When you multiply this across teams and tools, all this testing becomes increasingly unmanageable.

Major security activities occur at all stages of the SDLC. This includes

- Threat modeling and risk analysis at the planning stage
- IDE tools that test and deliver remediation guidance as developers write code in real time
- The “big three”—static application security testing (SAST), software composition analysis (SCA), and dynamic application security testing (DAST)—during the build and testing phases
- Penetration testing and continuous dynamic testing in production

In the quest to secure the SDLC, many organizations have found themselves drowning in too many tests, too many results, and too many tools. Managing AppSec at enterprise scale takes a combination of tools, processes, and people to keep development moving at the speed your business needs, while ensuring that you’re protecting your business, your reputation, and your customers.

Let’s take a look at a few of the important AST activities that are performed in each phase of the SDLC, beginning with at the heart of it all: risk management.

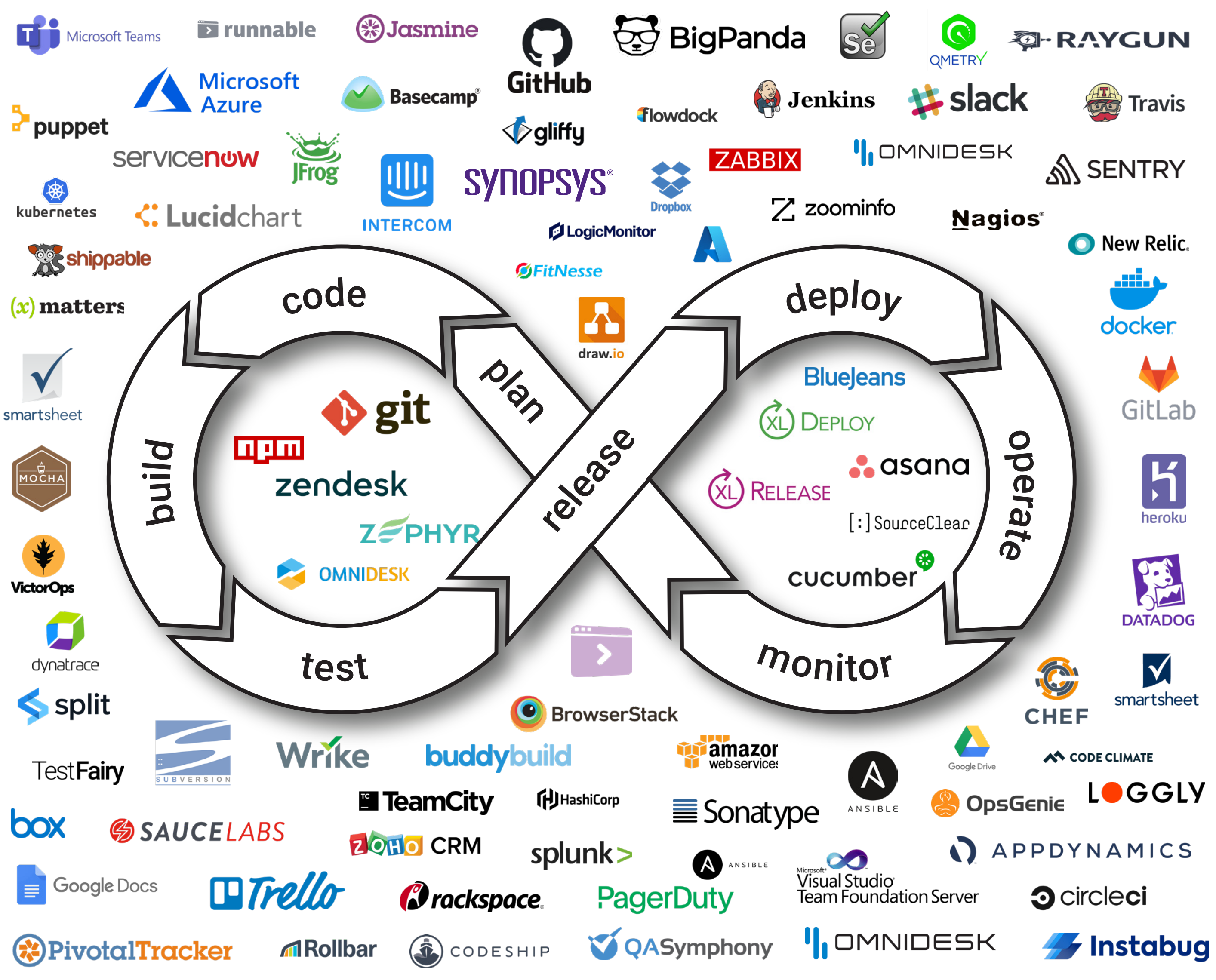
In the quest to secure the SDLC, many organizations have found themselves drowning in too many tests, too many results, and too many tools.

Software complexity means security complexity

Navigating complicated software environments can divert valuable resources away from key development activities, impeding agility and making it harder for organizations to respond quickly to changing market and customer demands.

On the security side, the increasing complexity of the security tooling landscape can slow down production and increase risk. When development pipelines get bogged down, development teams may wind up skipping or ignoring security gates to meet their development milestones. The problem is ubiquitous. A recent paper by the Enterprise Strategy Group, "[Cracking the Code of DevSecOps"](#) discovered that over 70% of enterprises are using more than 10 AST solutions.

Tool sprawl in the SDLC



Lack of standardization in DevOps tooling makes automating and integrating them into CI/CD pipelines a challenge for enterprise security teams.

Security complexity is a microcosm of the problem that software complexity poses across organizations. More security tools lead to more tests, which translates to more results. Since results are being returned from a variety of point tools, too often, developers receive duplicate results and inefficient, noncontextual remediation guidance. This means they waste valuable time and resources trying to triage security issues before they can even hope to start fixing them. Since the primary job of most developers is to ship software, not to ship secure software, they end up releasing software that they know is not secure or that has vulnerabilities. The effort required to manage tools, perform maintenance, and integrate tools into existing environments inhibits the organizations from remaining productive. This security tool sprawl adds complexity, drains resources, and has become unmanageable at scale.

Application security posture management gives you control

To rein in security sprawl, you need a comprehensive strategy that maps out your application security program. It should include your risk appetite and the policies that will enforce your SLAs, your critical testing needs, and the requirements for tools and vendors to help get you there. Ideally, this strategy will allow you to consolidate your tools, tests, and results and centralize activities like policy, test orchestration, and reporting.

Application security posture management (ASPM) tools can help you achieve this goal by providing a single source of truth to set and enforce policies, and identify, correlate, and prioritize security vulnerabilities across your SLDC. ASPM tools can act as a layer of abstraction between your application security team and your development team. They provide a central point of control to manage your AppSec program and remove developers from the complexity of the security tools themselves. This is the developer's dream—they don't aspire to learn the AppSec management tools, they want to consume relevant security information via the tools they are already using. ASPM tools allow AppSec teams to implement and manage consistent workflows and policies across an organization, without needing to replicate those policies across multiple tools and teams, or train development teams on how to use them.

Once this is in place and your AST tools are integrated, you can simplify issue interpretation, triage, and remediation with ASPM solutions that correlate and analyze data from a variety of sources. ASPM solutions also administer and orchestrate security tools so you can implement automated security policies. ASPM allows security teams to centrally manage application security findings by leveraging a consolidated view of security and risk status across the entire software development environment.

ASPM tools allow you to do this by providing a single unified view of your entire security landscape. This allows you to normalize, aggregate, and prioritize findings, assets, stakeholders, and policies in one centralized location. This consolidated approach to insight management improves security processes, optimizes resource allocation, provides a rapid time to audit to ensure regulatory compliance, and leads to a more robust and effective security posture across your organization.

Once you have this unified view, you can use the data to break down silos across your security tools and teams, facilitating continuous and efficient AppSec practices. Instead of juggling a dizzying array of tools, you can centralize your security efforts on a single platform, enabling consistent security assessments and streamlined workflows. This not only saves valuable time and resources but ensures that all stakeholders have access to a unified view of application security, leading to better decision-making and risk management.

Instead of juggling a dizzying array of tools, you can centralize your security efforts on a single platform, enabling consistent security assessments and streamlined workflows.

With the layer of abstraction in place and armed with the data you need to quickly assess your efficacy, you can consolidate the tools underneath. The impact of doing this will not only remove complexity and potential duplications in your security tool stack, but it will streamline budgets, training, support, and implementation to improve your overall appsec program TCO.

ASPM enables consistency in the implementation of your AppSec program, rapid risk insights, efficient prioritization of critical defects, and tool consolidation during a time of enormous budget and resource pressures.

So how do organizations increase security at the various stages of development? Let's look at a few best practices to help strengthen your security posture across the SDLC.

SDLC planning phase

AppSec is often overlooked in the planning phase of the SDLC, despite data that shows nearly [50% of your software system's "implementation bugs" can be attributed to design flaws.]() A secure software design helps prevent security breaches that stem from design flaws. Hence, comprehensive threat modeling is an essential exercise in the planning phase of your SDLC.

Threat modeling begins with conducting interviews with business owners of the software system to better understand the security risks that impact the business goals of the system. Major components, assets, threat agents, and security controls that exist in the system are then illustrated in the form of an architecture diagram to capture these entities and the relationships between them. The final step is to identify software-based risks and prioritize them according to business impact (e.g., unauthorized access to data or service availability).

Threat modeling early in the planning and design phase enables you to avoid costly rework to address security defects found later in the SDLC. Most importantly, finding, and remediating security problems earlier in the SDLC is less expensive, invasive, and time-consuming than waiting until code is written or QA tests are performed.

50% of your software system's "implementation bugs" can be attributed to design flaws.

SDLC coding/building phase

Static application security testing (SAST) is an integral part of AppSec. It is a thorough analysis of application source code in a nonruntime environment and without executing the code. SAST detects software security vulnerabilities and code quality issues early in the SDLC, when code changes are least disruptive and easiest to resolve. SAST tools should include

- Integrations with common developer workflows through IDEs, SCM systems, and CI pipelines, and that can trigger automated scans through each phase of the SDLC
- Configurable security and quality checkers that can be tuned to match the risk profile of each application
- The ability to run policy-based scans that align with your overarching AppSec strategy, while balancing both productivity and security throughout the SDLC
- Best-in-class identification of code quality issues along with the ability to track and manage compliance with common security and industry standards over time

With ever-increasing reliance on open source code, an SCA tool is essential in your modern AppSec arsenal. A good SCA tool helps manage security, license compliance, and code quality risks that come from the use of open source in applications, containers, and infrastructure-as-code (IaC). A best-of-breed SCA tool should

- Provide multiple methods for identifying open source based on which phase in the SDLC it's being run and the criticality of the software being tested
- Generate and monitor a complete Software Bill of Materials
- Include a comprehensive knowledgebase of open source components and supports its own security feed for enhanced and timely information and remediation guidance for open source vulnerabilities
- Quickly and easily analyze vendor-supplied binaries to identify weak links in your software supply chain without access to the source code

Another often overlooked AppSec test to consider running in the coding/building phase is malicious code detection (MCD). No one wants to believe that internal hacking will happen to them, but the truth is that disgruntled developers can and do plant malicious code in software systems. A thorough MCD test finds suspicious constructs, and identifies the malicious code that typical security tools can't find, along with the insider threat actors. MCD provides expert advice on malicious code management and delivers vulnerability remediation strategies.

SDLC testing phase

In addition to SAST and SCA being run in the testing phase where appropriate, interactive application security testing (IAST) is crucial in the testing phase of the SDLC. It utilizes runtime testing techniques to identify security risks associated with vulnerabilities in running web applications. IAST works through software instrumentation to monitor an application as it runs and gather information about what it does and how it performs. It is known for providing accurate results for fast triage. An IAST tool should provide

- Support for cloud-based applications (in addition to on-premises), microservices, containers, serverless, etc.
- Comprehensive test coverage with API discovery, tracking, and data flow map of your application and microservices
- The ability to track sensitive data and secrets—especially important for PCI DSS, GDPR, and other compliance standards

SDLC release/deploy phase

Penetration (pen) testing is an essential complement to automated AppSec testing. It examines application vulnerabilities and tries to exploit them using both automated testing tools and manual testing techniques to ensure that false positives are eliminated, and then delivers a detailed vulnerability report along with expert remediation advice. This allows your team to prioritize the most critical vulnerabilities for mitigation.

The true test of the security of an application is how it stands up to an attack in production. This is why you need to be doing continuous dynamic security testing in production to protect the most common entry point of malicious actors: web apps. You need a solution that tests your software using the same methods threat actors use when they are trying to breach them, without any impact to production. A dynamic testing tool should provide

- Automatic detection and analysis of code changes to web applications
- Alerts for newly discovered vulnerabilities (like Log4J)
- A team of security experts that focus on writing and deploying tests to ensure that the most up-to-date exploit information is included in every scan
- An unlimited number of websites and applications onboarded and scanned concurrently; often a cloud-based solution is preferred as it simplifies implementation and helps you scale fast
- Asynchronous testing, so teams don't have to wait for a scan of an organization's entire web app ecosystem to finish running before starting a test for a single feature or app
- No production impact

As organizations evolve from DevOps to DevSecOps, the challenge to shift everywhere means distributing security responsibility across the SDLC.

Simplify, automate, and shift everywhere to manage AppSec at enterprise scale

As application security testing has become a crucial component of software development in every sector, scaling security remains a challenge as software, threat, and regulatory landscapes grow and budgets remain flat or shrink. As organizations evolve from DevOps to DevSecOps, the challenge to shift everywhere means distributing security responsibility across the SDLC. The key means of effecting this transition are

- Streamlining AST to reduce costs and increase efficiency. Most organizations are using multiple AST vendors which multiplies the costs, both in developer time and in the resources required to implement, execute, and support these disparate solutions. Streamlining your tools and vendors will help rein in the overwhelming number of tools, tests, and results that your teams struggle to navigate.
- Developing a comprehensive strategy, plan, and policies. By doing an inventory of where your program stands today you can identify the ways it needs to improve, and make a plan and a timeline to implement those improvements. When teams work from well-defined policies they can enable consistent development at the velocity the business needs while maintaining security standards.
- Centralizing policy management, test orchestration, results prioritization, and risk insight. By using an ASPM tool to centralize and connect security data, software resources, policies, and insights, your organization can make quick, informed decisions to immediately bolster your security posture.
- Leveraging tools that enable you to secure software at scale. Equipping your teams with the tools and the external resources they need is key to building secure, high-quality software at the scale your business requires. Consolidating those tools and managing them with an ASPM tool will allow your security teams visibility across your entire enterprise landscape, without placing the added burden of tool sprawl on development.
- Maintaining continuous testing to manage risks in production. Ensuring the continuous security of your live apps and building effective mechanisms to identify and address active vulnerabilities present in your running applications is a crucial piece of security at the end of your SDLC. Malicious actors breach apps from the outside, so it's important to remember and plan for security that doesn't end when an application is released to production.

How Black Duck can help

Black Duck is the only vendor with a comprehensive portfolio of best-of-breed solutions that cover every stage of the SLDC. Black Duck AST solutions provide an open ecosystem, enabling organizations to leverage their existing AST tools while improving their risk posture and development productivity. Because Black Duck can aggregate and prioritize findings from 135+ Black Duck, third-party, open source, and other testing types like threat modeling and penetration testing, your organization can gain a consolidated, clear view of your software risk at any point in time.

Black Duck AST solutions empower organizations to run the right test, at the right time, at the right depth of analysis, with best-in-class solutions for the "essential three" AST testing types: SAST, SCA, and DAST. Black Duck also offers both on-premises security management solutions in Software Risk Manager™, and SaaS security management solutions via the Black Duck Polaris™ Platform and Continuous Dynamic™. And for those who need to outsource expertise, the Black Duck services team has over 500 security professionals ready to bridge the gap in your existing AST solutions.

The Black Duck is an industry leader you can trust, one that has been named the Leader in the Gartner® Magic Quadrant™ for Application Security Testing for the past seven consecutive years.

About Black Duck

Black Duck® offers the most comprehensive, powerful, and trusted portfolio of application security solutions in the industry. We have an unmatched track record of helping organizations around the world secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies. As the recognized leaders, experts, and innovators in software security, Black Duck has everything you need to build trust in your software. Learn more at www.blackduck.com.