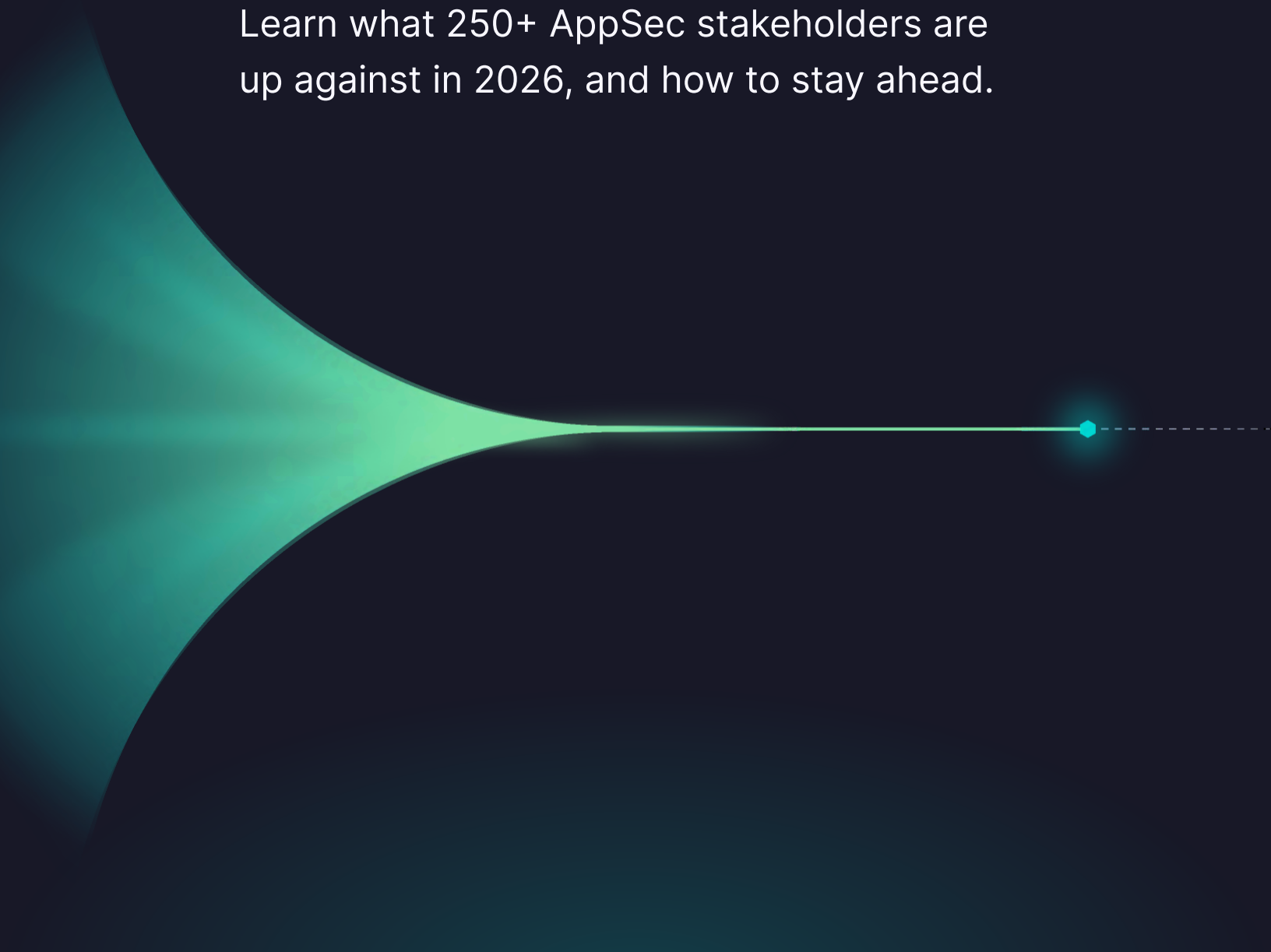
STACKHAWK

The 2026 AppSec Leader's Guide to Survival in the AI Era

Learn what 250+ AppSec stakeholders are up against in 2026, and how to stay ahead.



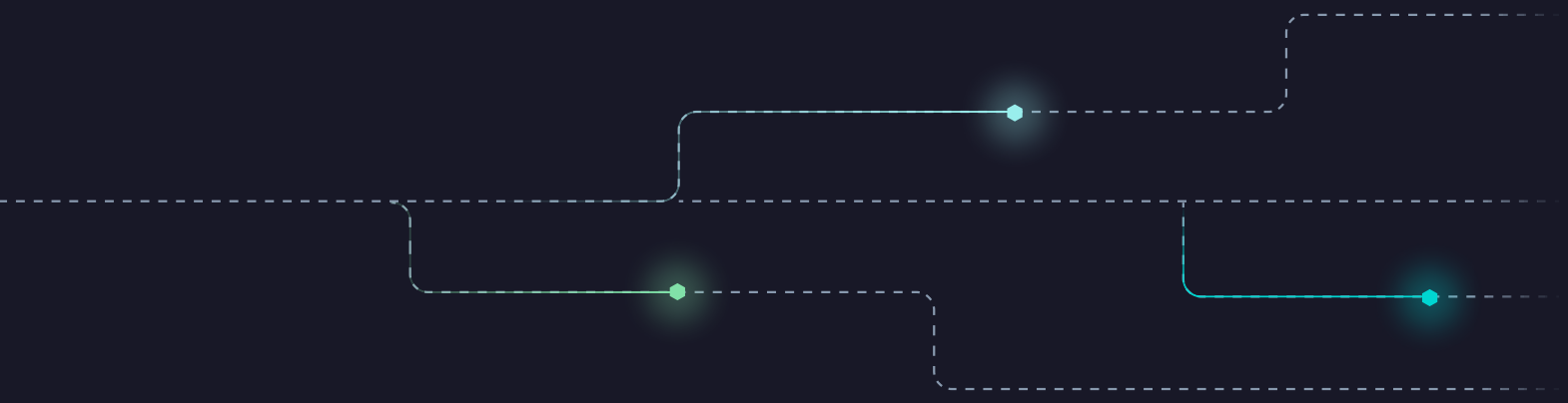
Introduction

AI has fundamentally changed how applications are built, and how they need to be secured.

Development velocity is 5-10x faster. Attack surfaces are growing faster than teams can track. Yet most AppSec programs are still operating with a playbook designed for a world that no longer exists.

We surveyed 250+ AppSec stakeholders to understand how teams are adapting. The findings confirmed what we suspected: teams are struggling to keep up with AI-powered development velocity, drowning in alerts from fragmented tools, and juggling new LLM/AI risks—all while facing increased pressure to prove program effectiveness to executives asking harder questions.

This guide combines proprietary research with practical guidance for AppSec leaders navigating this shift: what's changed, why traditional approaches are breaking, and how to build programs that actually scale with modern development.



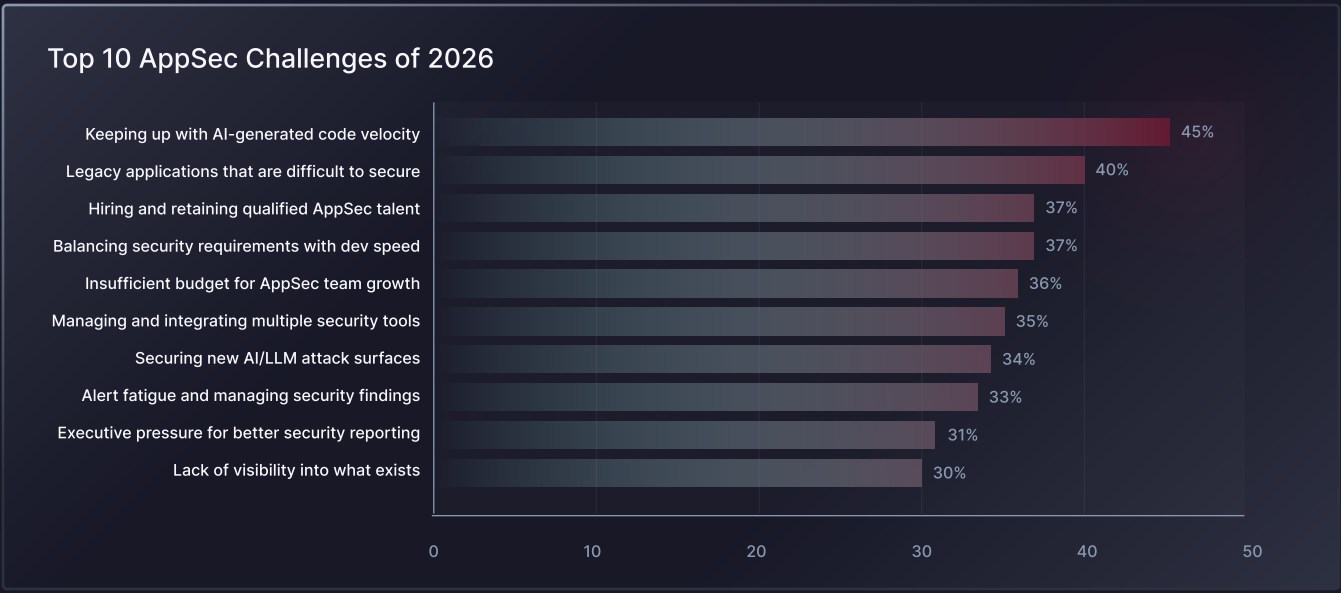
What's Inside

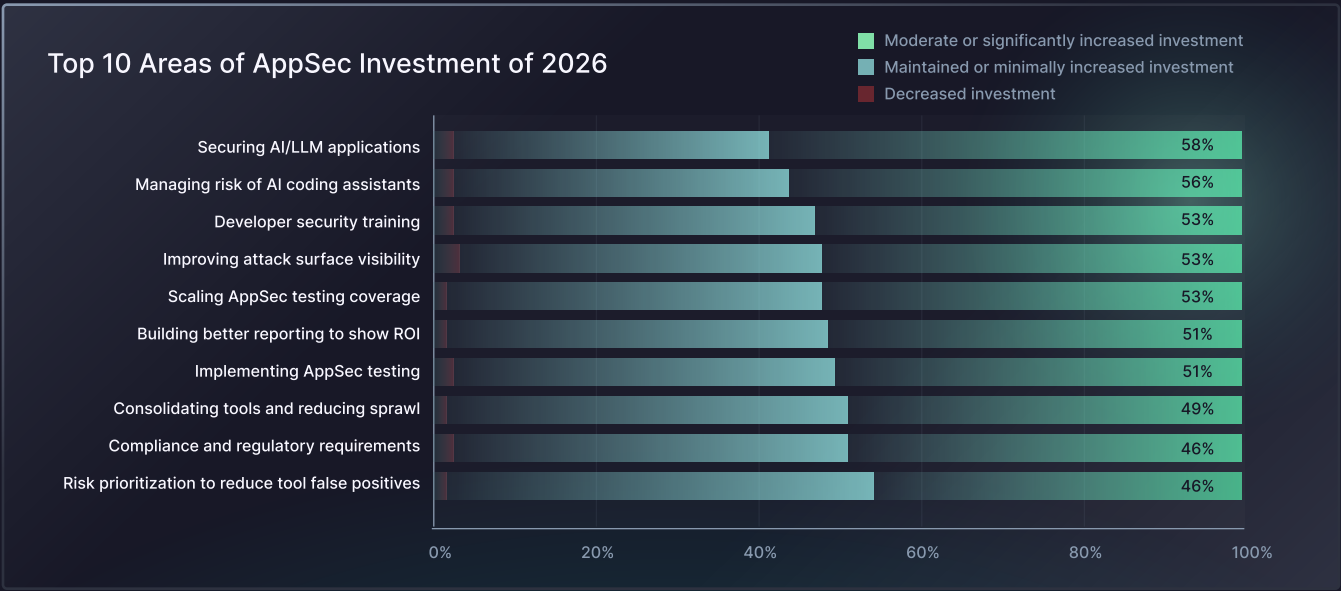
Survey Overview

This survey was conducted over two weeks in December 2025 with 259 respondents across application security stakeholders. Key demographics:

- **Industries:** Technology (56%), manufacturing (13%), and financial services (7%) were the top sectors represented
- **Roles:** 71% are decision makers over AppSec; the remainder are involved but not primary decision makers
- **Company size:** The majority (72%) have between 251–5,000 employees
- **AppSec maturity:** 84% have dedicated AppSec teams of 4+ people

The primary goal of the survey was to understand the challenges and initiatives shaping AppSec programs in 2026 and to translate those findings into actionable guidance. At a high level, here's what we found:





PART 1

Our New AppSec Reality: AI Has Changed Everything

TL;DR

AI has already reshaped how applications are built and AppSec programs are scrambling to catch up. Our survey of 250+ AppSec stakeholders reveals how quickly the landscape has shifted:

87%

87% of organizations have adopted AI coding assistants to some extent, with 35% at widespread or full adoption

77%

77% are building LLM/AI components into applications, introducing entirely new vulnerability classes

#1

"Keeping up with rapid development velocity and AI-generated code" is the #1 significant challenge for 2026

73%

73% are regularly asked by executives about their organization's application attack surface or risk posture

AI Code Assistants Are Now The Norm

GitHub Copilot. Cursor. Claude Code. Pick your flavor.

AI coding assistants have raced past "emerging technology"—87% of organizations we surveyed have adopted them to some extent, with over a third at widespread or full adoption. What was experimental two years ago is now standard, which means more code shipping at exponentially faster speeds.

More Code, More Complexity

Developers using AI assistants are producing 5-10x more code than they did two years ago—a shift from writing code to reviewing, iterating, and accepting it. The result: more applications, more APIs, more microservices, more distributed systems.

The attack surface is exploding.

And the complexity compounds. A modern application isn't a monolith you can point a scanner at—it's dozens of services, hundreds of APIs, and thousands of endpoints, all evolving independently. Each new component is another entry point, another integration, another thing to secure.

Less Context, Less Visibility

When developers write code line-by-line, they develop intuition about how it works, what it touches, where the edge cases live. They understand authorization rules because they implemented them. They know data flows because they traced them.

AI-assisted development changes that equation. Developers might accept complete API implementations without deeply understanding the business logic. They review for functionality, not security implications. The code works. The tests pass. Ship it. The context gap grows with every AI-assisted commit.

Discovery Can't Keep Pace

Shadow applications multiply faster than teams can track. Manual discovery methods (spreadsheets, quarterly surveys, parsing cloud bills) are perpetually weeks if not months behind reality. By the time you document what exists, developers have already shipped ten more services.

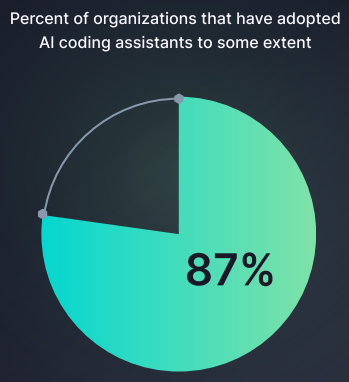
You can't secure what you can’t see. And right now, most AppSec teams don’t have that visibility.

Survey Stat

AI Coding Assistant Usage & Risks

87% of organizations surveyed have adopted AI coding assistants such as GitHub Copilot, Cursor, or Claude Code to some extent. An astounding **35%** are already at widespread or full adoption. But that productivity boost comes with risk.

Over half of respondents view the use of AI coding assistants as a moderate or significant risk to their organization, and “keeping up with rapid development velocity and AI-generated code” was the number one significant challenge cited by AppSec stakeholders.



AI/LLM Components Deeply Embedded Into Applications

Not only are we pumping out code with AI, but we're building AI/LLMs directly into applications. RAG systems for customer support. AI-powered search. Intelligent agents that take actions on behalf of users. Features that would have required months of ML engineering work now ship in weeks using off-the-shelf LLM components.

LLM Components Are Everywhere

These aren't sandbox experiments—they're production systems interfacing with customer data and core business functions. And unlike traditional application logic, LLMs are non-deterministic. The same input can produce different outputs. Behavior changes based on context, prompt construction, and model updates you don't control. This makes them harder to test, harder to predict, and harder to secure.

New LLM Risks

The result is a rapidly expanding attack surface and an entirely new set of risks:

- Prompt injection attacks that manipulate LLM behavior to bypass business logic
- Data leakage through context windows that inadvertently expose sensitive information
- Context poisoning where attackers corrupt the knowledge base feeding the AI
- Guardrail bypasses that trick models into performing unauthorized actions

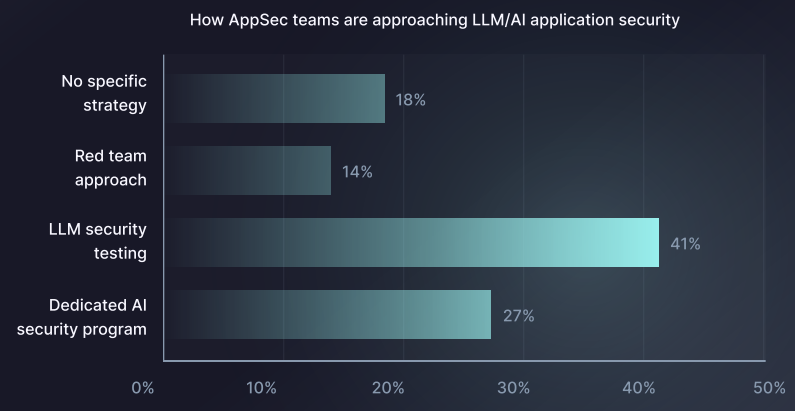
Legacy AppSec tools weren't built for this. Static analysis can't detect prompt injection vulnerabilities or validate whether your RAG system properly segregates customer data. These risks only manifest at runtime, when the LLM is actually processing requests, retrieving context, and executing actions—which means they require runtime testing to catch.

Survey Stat

AI Application Security Strategy

"Understanding and securing new AI/LLM attack surfaces" was the **2nd top cited** challenge for 2026 and 77% of surveyed organizations are building LLM/AI components into applications.

82% of organizations are thinking about how to secure LLM/AI components.



More Risks, Exploited Faster

AI didn't replace your existing security challenges. It added to them.

Old Vulnerabilities, New Scale

Traditional vulnerability classes—injection attacks, broken authentication, access control failures—haven't gone away. They're showing up more frequently because there's simply more surface area for the same mistakes. And the risks that actually cause breaches—authorization bypasses, business logic flaws, API security gaps—are proliferating as developers reviewing AI-generated code ask "does this work?" not "what happens if someone manipulates this workflow?"

Exploitation Timelines Are Compressing

Attackers aren't waiting for you to catch up. What used to take weeks from disclosure to active exploitation now happens in days—sometimes hours. Automated reconnaissance finds exposed APIs and forgotten endpoints faster than manual discovery ever could. AI-assisted attack tooling is lowering the barrier for weaponization.

The math is brutal: more vulnerabilities, across a larger attack surface, being exploited faster, with the same team size you had three years ago.

AppSec Talent Gap is Widening

We all know that AppSec professionals are typically outnumbered by developers (somewhere in the 1:50 range). That hasn't changed, but the job has.

AppSec used to be about running scans and triaging findings. Now it requires understanding cloud-native architectures, API security, CI/CD pipelines, AI/LLM components, and the developer workflows that ship code 5-10x faster than they did two years ago. The role has evolved from security specialist to a hybrid that spans development, security, and platform engineering. That's an incredibly specialized skill set—and it's in short supply.

Our survey confirms it: hiring and retaining AppSec talent ranked among the top challenges for 2026. Organizations aren't just competing for a limited talent pool; they're competing for people whose skills need to evolve as fast as the threat landscape.

You can't hire your way out of this—but you also can't keep asking the same team to absorb exponentially more work with the same playbook.

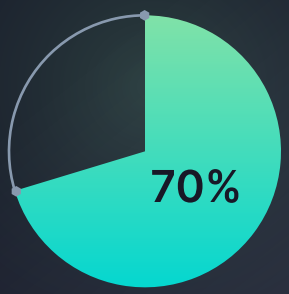
Survey Stat

The Job Is Changing, The Talent Pipeline Isn't

Over 70% of respondents cited hiring and retaining AppSec talent as at least a moderate challenge for 2026. It's not hard to see why: the role has expanded faster than the talent pipeline.

Finding engineers who understand cloud-native security, API vulnerabilities, and AI/LLM risks—while also speaking developer—is increasingly difficult.

Percent of respondents that cited hiring and retaining AppSec talent as a challenge for 2026



Boards and Executives Demanding Answers

AI risk is the boardroom topic of the moment. But here's what most executives haven't fully grasped: when they're assessing AI risks, they're really assessing software risks. And the only team who actually understands and knows how to reduce that risk is AppSec.

This is putting AppSec at the center of organizational risk management in ways it never has been before. Post-SEC disclosure rules changed the game. Board-level security oversight is now a regulatory requirement. Boards are asking harder questions. Executives expect real answers:

- "What's our actual attack surface?"
- "How is risk trending over the last quarter?"
- "What's our actual application risk posture?"
- "Are our security investments paying off?"

Our survey found that 73% of AppSec stakeholders are regularly asked by their board or executive leadership about application attack surface or risk posture. The attention and prioritization is there.

The Reporting Gap

The problem is most AppSec leaders can't answer these questions with confidence. They can report activity—"we ran 10,000 scans this month"—but they can't connect that activity to risk reduction. They can show dashboards full of findings, but they can't explain which applications pose the most business risk or whether the vulnerabilities they're fixing actually matter.

The gap between what executives need to know and what AppSec programs can currently report has never been wider. It's not a failure of effort. It's a failure of tooling and process that was never designed to answer risk-based questions.

Closing that gap isn't optional. Boards are personally accountable for cybersecurity oversight. Regulators are paying attention. AppSec leaders need to be able to answer the hard questions—and that requires a different foundation than most programs have today.

In Summary

AI development has changed what "comprehensive AppSec" means. The old playbook—the one optimized for humans writing 100 lines of code per day—doesn't match the new reality. Here's what we're facing:

- AI coding assistants mean more code, faster and attack surfaces expanding faster than teams can track.
- AI/LLM components in production introduce new vulnerability classes that traditional tools can't detect.
- Old risks are scaling while exploitation timelines compress from weeks to hours.
- The AppSec role evolved faster than the talent pipeline—and you can't hire your way out.
- Boards are asking questions about risk posture that most teams can't confidently answer.

PART 2

The AI-Era Impact: When AppSec Can't Keep Pace

TL;DR

When the landscape shifts this fast, something breaks. Teams are absorbing exponentially more work with the same resources and the same playbook. Our survey reveals where AppSec programs are feeling the strain:

50%

50% of AppSec teams spend more than 40% of their time just triaging and prioritizing findings

71%

71% cited alert fatigue as a moderate to critical challenge for their AppSec team in 2026

30%

Only 30% of organizations are very confident in their understanding of their application attack surface

94%

94% use at least one AppSec testing tool, yet production risks still slip through

Prioritization Challenges

When developers shipped code at human speed, AppSec teams could maintain a reasonable mental map of what existed and what mattered. AI code assistants have broken that model. Apps, APIs, and services now spin up faster than manual process can track.

The Visibility Problem

And without visibility into your attack surface, you're working from an incomplete inventory and testing the applications you know about, while critical systems remain completely unprotected.

The Prioritization Problem

Even when you find vulnerabilities, prioritization becomes the next challenge. Your scanner dashboards show hundreds of "critical" and "high" severity findings. Which actually pose business risk? Without context—which apps process sensitive data, how they connect to revenue-generating systems, which vulnerabilities are exploitable—you can't answer that question.

When everything is urgent, nothing is prioritized. Finite remediation resources get allocated based on who yells loudest, not what reduces the most risk.

Wasted Time Triaging

AI code assistants didn't just accelerate feature delivery—they multiplied the volume of security alerts that AppSec teams have to process.

SAST tools generate thousands of findings per app. Industry research shows 95-98% are unexploitable, including false positives, duplicate findings, vulnerabilities in unreachable code paths, and theoretical issues that don't manifest in practice.

The Math Doesn't Work

Same team size. Same hours in the day. But 5-10x more code generating 5-10x more findings. One AppSec engineer triaging for 50 developers was already unsustainable—now those same developers produce five times the alerts.

The result is AppSec engineers spend huge chunks of their time on low-value work:

- Manually analyzing findings
- Determining reachability
- Building proof-of-concept exploits
- Documenting acceptable risk

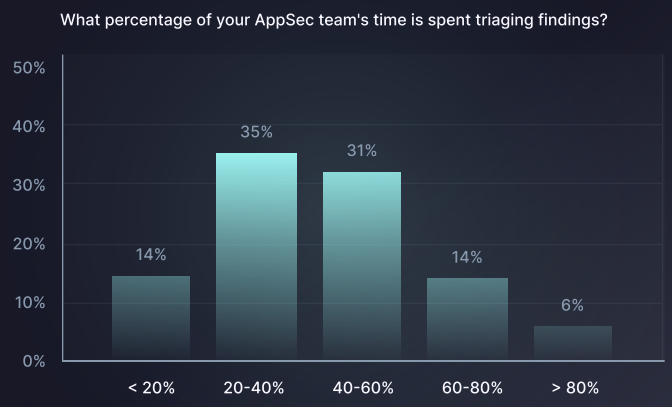
You can't triage your way out of this. The volume will always exceed capacity.

Survey Stat

Triage Challenge

Half of surveyed AppSec stakeholders spend more than 40% of their time triaging findings from tools.

On top of that, **71%** cited “alert fatigue” as a moderate to critical challenge for 2026.



Production Risks Slip Despite Many Tools

Orgs aren't under-investing in security tooling. Yet the vulnerabilities that cause actual breaches keep slipping through.

Maybe AppSec teams are too busy validating whether findings are real, or maybe it's because their tooling isn't equipped for AI-era risks.

Look at the last five major breaches. What caused them? Authorization bypasses. Broken authentication. Business logic flaws. API security gaps. These are all vulnerabilities that only manifest when applications are actually executing, processing requests, and enforcing (or failing to enforce) business rules.

SAST fundamentally cannot answer runtime questions. It analyzes code, not behavior. Pen tests can, but quarterly isn't fast enough for the pace of AI. WAFs might block known attack patterns, but they can't understand your business logic or detect when authorization is broken by design. And testing in production means the vulnerability is already exposed to attackers.

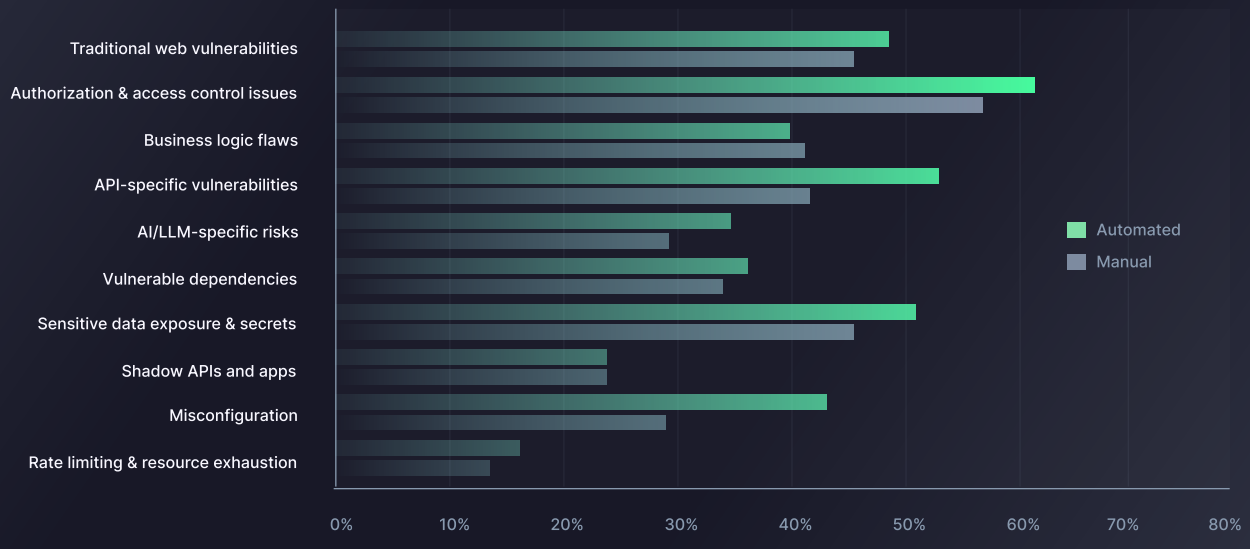
The result: AppSec teams stay busy with massive volumes of findings while the exploitable risks that actually cause breaches reach production undetected.

This isn't a failure of effort. AppSec teams are working harder than ever. It's a gap in approach.

Survey Stat

What AppSec Teams Are Testing For

94% of organizations use at least one AppSec testing tool category, with the majority using 2 or more. These are the top risks AppSec teams are focused on finding, through automated vs. manual methods:



Activity Reporting Won't Cut It

More than ever, boards expect clear answers about risk. What they're getting instead are activity metrics that don't actually answer their questions.

When CISOs can't demonstrate program effectiveness, they can't justify budget increases. When they can't show which investments reduced the most risk, they can't make data-driven resource allocation decisions.

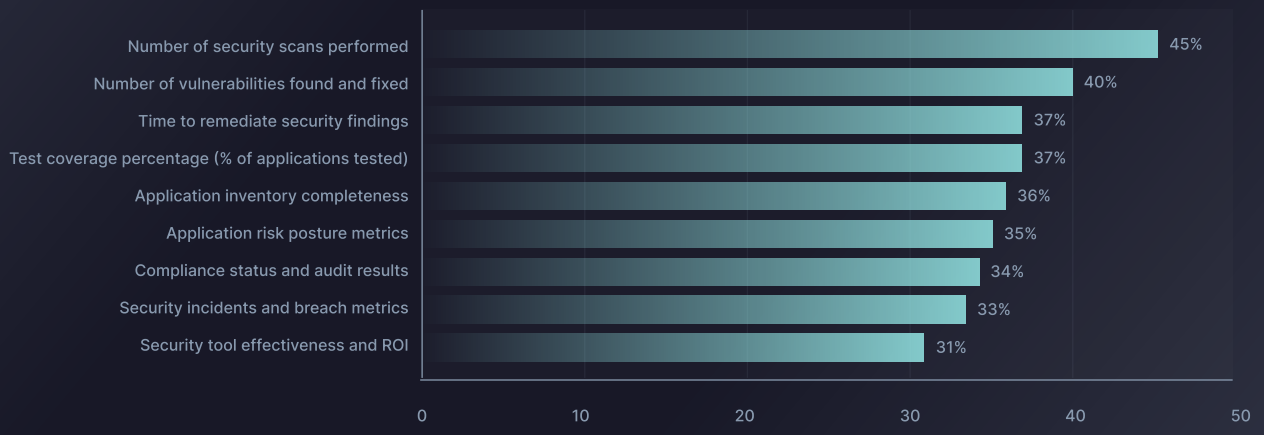
When they report activity instead of outcomes, executives question whether AppSec spending is actually making the organization more secure.

The gap between what boards need to know and what AppSec programs can report has never been wider—especially when boards are personally accountable for security oversight and SEC enforcement is active.

Survey Stat

Top Metrics Reported to Executives and Boards in 2026

Our survey asked AppSec stakeholders which metrics they'll be expected to report to executives in 2026 and the results confirm the disconnect:



In Summary

The current approach isn't scaling. Teams have more tools, more scans, more findings—but no better answers to the questions that actually matter. Here's where it's breaking down:

- Without visibility into the full attack surface and tons of noise from tools, teams can't prioritize what actually matters.
- Alert fatigue is real. 50% of teams spend more than 40% of their time triaging findings, most of which aren't exploitable.
- Risks slip through despite heavy tooling investment because the vulnerabilities causing breaches only manifest at runtime.
- Activity metrics don't answer executive questions. Boards want risk posture, not scan counts.

PART 3

What AppSec Leaders Actually Need: Intelligence

TL;DR

We have more tools, more scans, more findings, but no better answers to the questions that actually matter. AppSec teams are drowning in data without the intelligence they need to provide the answers boards are demanding.

By intelligence, we mean:

Security findings connected to business context

Application risk posture tracked over time

Remediation efforts mapped to actual risk reduction

Intelligence is having the right information, structured in ways that enable decisions. It determines whether your entire security program is working on things that reduce risk or just things that generate activity.

Imagine a 2026 where AppSec programs are built on intelligence, not guesswork. Where leaders can confidently answer the hard questions. Where resources flow to the highest risks. Where security work connects to business outcomes.

That's our vision at StackHawk. Here's what that looks like in practice.

Know What You Have

Manual inventorying was already struggling to keep up, but in 2026, there is no way around having a complete, accurate attack surface inventory. Not the spreadsheet that's six months out of date. Not the partial view from your cloud bill. The actual, comprehensive picture of every application, API, and service running in your environment.

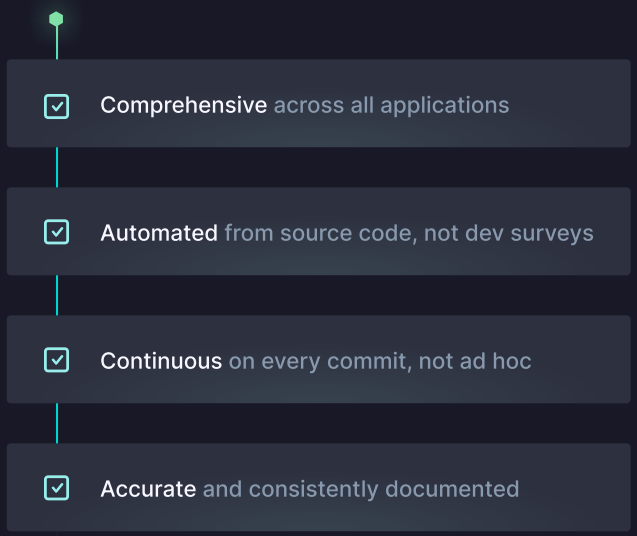
Continuous, Not Point-in-Time

This requires continuous discovery, not point-in-time surveys or documentation a human has to remember to update. You need automated systems that map your attack surface from source code and detect new applications the moment they're deployed.

Comprehensive; Source of Truth

You need visibility into everything. The POC that became production. The API someone exposed for a partner integration. The microservice the team forgot about. The AI/LLM components embedded in customer-facing features.

You can't secure what you don't know exists. Intelligence starts with knowing what you have.



Know What's Tested and How

Coverage metrics are meaningless if they're measured against an incomplete inventory. 90% coverage sounds impressive until you realize you're testing 90% of the 40% of applications you know about and your actual coverage is 36%.

Coverage With What?

Even accurate coverage numbers don't tell the whole story. SAST catches code-level issues but can't validate runtime behavior. DAST proves exploitability but requires running applications. IaC scanning catches misconfigurations before deployment. SCA flags vulnerable dependencies. Each approach has strengths and blind spots.

Match Testing to Risk

Intelligence means knowing not just what's tested, but whether you're testing it the right way. Your customer-facing payment API needs runtime testing that validates authorization actually works—SAST alone won't cut it. Your infrastructure-as-code needs scanning before it deploys misconfigured resources. Your AI/LLM components need testing that can detect prompt injection and guardrail bypasses.

Intelligence tells you which applications need which types of coverage—and where you have gaps that matter. You can prove to executives that testing investments are covering what actually matters. Coverage shifts from vanity metrics to strategic tooling.

Know How Resources Are Allocated

AppSec resources—headcount, tooling spend, vendor services—should flow to where they reduce the most risk. In practice, they're usually distributed based on organizational structure, historical precedent, or whoever asks loudest.

The Mismatch Is Getting Worse

With AI-assisted development multiplying output unevenly across teams, resource allocation is increasingly misaligned. The team shipping the most AI-generated code might not have proportional AppSec support. The applications integrating LLM components might be getting the same coverage as static internal tools.

Allocate Based on Data

Intelligence enables strategic resource allocation. Map headcount to risk instead of spreading it equally. Justify tooling spend with measurable risk reduction instead of feature checklists. Connect vendor spend to outcomes instead of contracted hours consumed.

Can you answer these questions right now?

- **Which apps** pose the most business risk, and do they have proportional AppSec support?
- **Which security tool investment** reduced the most application risk last quarter?
- **What would happen to your risk posture** if you reallocated 20% of your budget?

Intelligence means you can answer these questions with data, not guesses.

Know What's Highest Risk

You have 1,500 high-severity findings. Do you know which ten actually matter?

Context-Driven Prioritization

Prioritization comes down to two dimensions: likelihood and impact. Likelihood factors in exposure (internet-facing vs. internal) and exploitability. Impact factors in business criticality and data sensitivity. Combine them and you get actual risk—not just scanner severity.

An authorization bypass in your payment API that processes customer financial data and is exposed to the internet is fundamentally different from SQL injection in an internal demo environment, even if both are rated "critical" by your scanner. A prompt injection vulnerability in a customer-facing AI chatbot is fundamentally different from one in an internal experimentation tool.

From Reactive to Strategic

Intelligence provides the context that transforms generic severity ratings into business risk scores. You prioritize based on what would actually harm your organization if exploited, not just what your tools flag as important.

Risk scoring that maps to business impact shifts AppSec from "fix whatever the scanner reports" to "fix what reduces business risk." From treating all critical findings as equally urgent to having confidence that the ten vulnerabilities your team is addressing this sprint are the right ten.

Know How Risk Is Trending

Point-in-time snapshots don't tell you if you're improving. A single assessment shows where you are today—but not whether you're gaining ground or falling behind.

You need data over time to understand:

- ☑ Is your attack surface expanding faster than your ability to secure it?
- ☑ Are security investments actually reducing risk, or just keeping pace with growth?
- ☑ Which initiatives moved the needle on application risk posture?
- ☑ Which initiatives consumed resources without measurable impact?

Trend data transforms how you evaluate your program. Instead of asking "how many vulnerabilities do we have?" you can ask "are we reducing risk faster than we're accumulating it?" Instead of defending tool purchases based on features, you can show which investments actually moved the needle.

Board-Ready Reporting

Intelligence enables reporting that executives actually care about. Not "we ran 10,000 scans" but "we reduced high-risk exposure by 40% this quarter." Not "we fixed 500 vulnerabilities" but "we eliminated authorization vulnerabilities in customer-facing APIs, reducing financial data exposure risk."

This is the difference between reporting activity and demonstrating value. Boards don't want to hear that security is busy—they want to know that risk is decreasing and that their investment is working.

Trend data proves program effectiveness. It gives CISOs credible answers when boards ask the hard questions—and credibility is what unlocks budget and organizational support for mature security programs.

☑ In Summary

Intelligence is the foundation that makes everything else work. Without it, you're testing blind, prioritizing by gut, and reporting activity instead of outcomes. Here's what intelligence enables:

- ✓ **Know what you have.** Continuous, automated discovery of your full attack surface.
- ✓ **Know what's tested and how.** Coverage measured against reality, with the right approach for each application.
- ✓ **Know how resources are allocated.** Spend flowing to where it reduces the most risk.
- ✓ **Know what's highest risk.** Prioritization based on likelihood and impact, not scanner severity.
- ✓ **Know how risk is trending.** Data over time that proves program effectiveness.

PART 4

The Playbook: Building Intelligence-First AppSec

Don’t worry, we’re not here to push product or tell you the only solution is to replace everything in your security stack. This playbook will guide you through the bricks we recommend laying to build your intelligence foundation. The order is important because each layer enables the next:

1. Attack Surface Visibility: Know What You Have
2. Runtime Testing: Surface Real Risks, Not Noise
3. AppSec Program Oversight: Prioritize for Maximum Efficiency
4. AppSec Intelligence: Measure Risk Reduction

For each layer, we’ll discuss...

- What’s Broken Today
- What's Needed in 2026
- Hot Takes
- How to Achieve It

1

Attack Surface Visibility:

Know What You Have

You can't secure what you don't know exists. Visibility is the foundation—without a complete picture of your attack surface, everything else is guesswork.

What's Broken Today

Current Discovery Can't Keep Up

Most organizations track their application attack surface using manual spreadsheets, developer surveys, or network traffic-based solutions. Manual approaches fail because they're outdated the moment they're created—especially with AI development where applications deploy 5-10x faster. Network traffic-based solutions fail because they only detect applications after they're in production and processing requests, meaning staging environments, dormant services, and newly deployed applications remain invisible until it's too late.

What's Needed in 2026

Continuous Discovery from Source Code

Attack surface visibility requires automated discovery that operates at the pace of development. Connect directly to source code repositories and detect applications, APIs, and services automatically, the moment they're committed.

Real-time mapping maintains accuracy without manual updates. New repository? Detected. New API specification? Mapped. New deployment? Tracked. Context layering gives every discovered asset risk-relevant attributes: exposure level, data sensitivity, business criticality, environment, and ownership.

Hot Take

> Almost every AppSec leader thinks they know their application attack surface. Then they connect StackHawk to their source control manager and find out that only knew about <50%.
>
> Joni Klippert, Co-Founder and CEO STACKHAWK

How to Achieve It

Building comprehensive visibility requires tooling that automates discovery, processes that validate and enrich what's found, and people who maintain accuracy over time.

People

Assign discovery ownership. Someone ensures integrations stay connected and context tags remain accurate. Not full-time, but it needs a clear owner.

Train teams on classification. Engineering will tag and validate discovered assets. They need to understand your taxonomy and why context matters for prioritization.

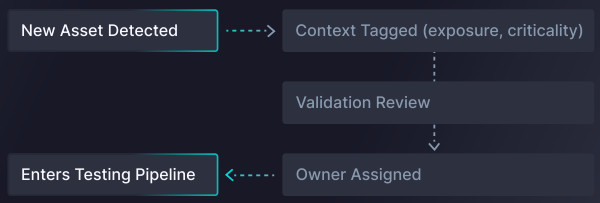
Establish application owners. Beyond just assigning names, create a culture of ownership. Owners are accountable for security decisions, remediation timelines, and risk acceptance.

Build a feedback loop. Discovery will surface unknowns. Treat these as collaboration opportunities—engineering validates, AppSec updates context, both address gaps together.

Process

Establish a validation cadence. Monthly reviews where AppSec and engineering validate discoveries and confirm context tags. Quarterly deep dives to audit the complete attack surface.

Define ownership and workflow. Every application needs a clear owner. Define what happens when discovery finds something new—does it enter testing automatically or require validation first?



Technology

Source-code-based discovery is non-negotiable. Choose tooling that connects to repositories and continuously scans for applications, APIs, and services. Look for:

- Auto-detection across GitHub, GitLab, and Bitbucket organizations
- API spec parsing (OpenAPI, GraphQL, gRPC) without manual configuration
- AI/LLM component and MCP integration identification
- Auto-generated specs from source code where documentation doesn't exist

Prioritize continuous discovery over point-in-time surveys. Set up automated discovery to run continuously—not quarterly, not monthly, continuously. New commits trigger discovery. New repositories are automatically detected.

Context emerges from what you discover. Source-code-based discovery doesn't just find applications and APIs. It should also surface important metadata and risk insights that can help inform your testing and remediation efforts. Metadata may include ownership and languages and frameworks in use. Risk insights may include commit activity within repositories, sensitive data (i.e. PII, PCI, PHI), and LLM interfaces.

2

Runtime Testing:

Surface Real Risks, Not Noise

Testing generates the findings that drive remediation. But if you're buried in false positives while real risks slip through, your testing strategy needs to change.

What's Broken Today

SAST First, DAST "Maybe Later"

The industry positioned static analysis as comprehensive security, with runtime testing optional. But SAST has fundamental blind spots—it cannot answer runtime questions about authorization, authentication, or business logic. The vulnerabilities actually causing breaches.

Legacy DAST didn't solve this because it wasn't built for modern development. Weeks of manual configuration, specialized engineers, testing production after code shipped. Too slow, too late, too expensive.

What's Needed in 2026

Runtime Testing First

AI development makes runtime testing more critical, not less. When developers accept AI-generated code they didn't write, they have less intuition about its security implications. Runtime testing is the safety net.

Runtime testing delivers exploitable findings immediately. 10-100x fewer findings than SAST, but every one is real. In the AI era, the challenge isn't catching syntax errors—it's verifying that code actually behaves securely at runtime.

Hot Take

> In the AI era, runtime testing has to be the #1 priority. Auth bypasses, business logic flaws, LLM vulns. Only DAST finds them and proves what's exploitable. Implementation complexity is no longer an excuse.
>
> Scott Gerlach, Co-Founder and CSO STACKHAWK

How to Achieve It

Effective runtime testing requires modern tooling built for CI/CD, processes that balance speed with rigor, and developers equipped to act on findings.

People

Train developers on runtime vulnerabilities. Authorization bypasses, authentication failures, and business logic flaws may be less familiar than SAST findings. Use real examples from your environment.

Build a security champions program. Developers interested in security become advocates, help others understand findings, and provide feedback on what's working and what's not.

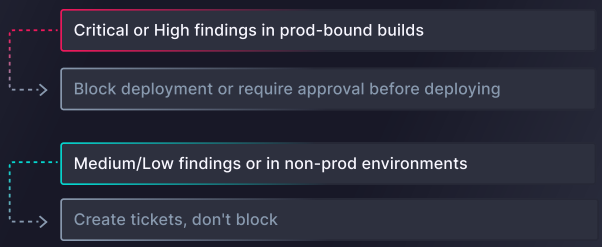
Create vulnerability playbooks. Document what each vulnerability class is, why it matters, how to fix it (with code examples), and how to prevent it.

Establish on-call for critical findings. When DAST finds a critical exploitable vulnerability in production, someone responds fast. Build a rotation for immediate triage.

Process

Phase your rollout. Start with 5-10 highest-risk applications. When developers see real exploitable vulnerabilities caught, they become advocates for expansion.

Define failure thresholds and SLAs. What blocks deployment vs. creates a ticket? Critical findings in production block; medium findings in staging create tickets. Document policies clearly and enforce them consistently.



Technology

Modern DAST is fundamentally different. Evaluate for:

API-driven architecture
REST, GraphQL, gRPC coverage with auto-generated tests from specs and modern auth support

CI/CD-native integration
Configuration-as-code, results in PR comments and tickets, no separate infrastructure

Low operational burden
Deterministic results, fast enough for every commit, no dedicated security engineers required

Developer-friendly remediation
Contextual guidance with code examples, not just vulnerability descriptions

LLM security testing is essential. If you're building AI features, you need runtime testing for prompt injection, data leakage, guardrail bypasses, and context poisoning. Static analysis is blind to these.

Bridge the gap between AppSec and developers. Developers should be able to run scans themselves, interpret results without security expertise, and validate fixes before pushing to production. But AppSec needs to be able to easily manage and report on results.

Prioritize broad and deep test coverage. Breadth means comprehensive protocol support plus native handling of modern authentication patterns. Depth means going beyond OWASP Top 10 to test for complex risks like authorization and business logic flaws.

3

AppSec Program Oversight:
Prioritize for Maximum Efficiency

Visibility tells you what exists. Testing tells you what's vulnerable. Oversight connects those findings to business context so you can prioritize what actually matters.

What's Broken Today

Siloed Tools, No Context

Most AppSec programs have visibility and testing operating in silos. Discovery data sits in a spreadsheet. SAST generates 15,000 findings. DAST produces 150. Each tool operates independently with no connection to business context.

Without connecting what's there (visibility) to what's vulnerable (testing), teams struggle to prioritize remediations and testing on what actually poses the most business risk and end up wasting cycles on low-impact work while high-risk vulnerabilities persist.

What's Needed in 2026

Visibility + Testing for Smarter Programs

Oversight emerges from combining visibility and testing into a unified view, then layering business context on top. Risk-based prioritization combines severity × criticality × exposure × exploitability. This transforms 1,500 "critical" findings into the 10 that actually matter.

When SAST and DAST both flag the same vulnerability, you triage it once. Cross-referencing reveals what's actually exploitable—DAST validates which SAST findings manifest as real runtime risks. Resource allocation finally aligns with actual risk instead of generic severity scores. You shift from "fix everything" to "fix what reduces business risk."

Hot Take

> More tools won't save you. Testing is a commodity, context is scarce. Connecting the two is how you will finally be able to allocate your time and budget based on business risk, not gut feel.
>
> Scott Gerlach, Co-Founder and CSO STACKHAWK

How to Achieve It

Unified oversight requires a platform that correlates across tools, processes that translate findings into prioritized action, and people aligned on how decisions get made.

People

Assign platform ownership. Someone maintains integrations, tunes scoring, generates reports, and ensures data quality to prevent blockages and keep the system accurate.

Train on risk scoring. The AppSec team needs to understand how scores are calculated, how to use them for prioritization decisions, and when to escalate based on thresholds.

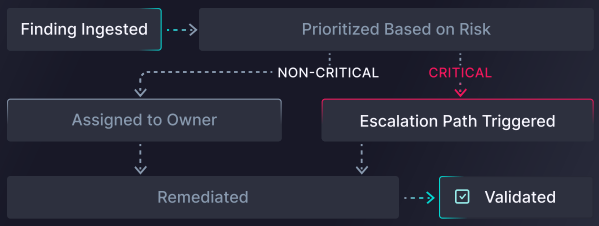
Educate stakeholders. Developers and PMs will question prioritization. "Why is this critical deprioritized?" Share the framework broadly with examples to prevent contentious conversations.

Establish executive reporting cadence. Monthly dashboard to executive team, quarterly deep dive with board, ad-hoc for critical incidents. Predictable reporting builds credibility.

Process

Document your risk scoring and prioritization model. How do you weight severity, criticality, exposure, and exploitability? Make it transparent so teams understand why findings are prioritized the way they are.

Create escalation paths. Critical vuln in prod? Define immediate notification, escalation timeline, emergency patch process, and post-incident review requirements.



Technology

Implement a platform layer that connects everything:

- **Discovery data**
  Complete attack surface inventory with context
- **DAST findings**
  Runtime vulnerabilities with exploitability proof
- **SAST findings**
  Static code analysis results
- **SCA findings**
  Third-party dependency vulnerabilities
- **Context metadata**
  Exposure, business risk, data sensitivity, ownership

Build or buy correlation capability. Identify and correlate when multiple tools report the same vulnerability (i.e. across code vs. runtime) and de-duplicate automatically. Manual correlation doesn't scale with AI speed.

Implement risk scoring. Combine vulnerability severity, asset criticality, exposure, and exploitability to understand real business risk. You need the capability to combine these dimensions automatically. The exact formula is customizable.

Create views for different audiences. Practitioners need detailed findings. AppSec leaders need coverage metrics. Executives need risk posture summaries. One platform, multiple views.

4

AppSec Intelligence:
Measure Risk Reduction

Visibility, testing, and oversight generate data. Intelligence transforms that data into answers—for your team, your executives, and your board.

⚠ What's Broken Today

Activity-Centric Metrics

Most AppSec programs report activity metrics that executives don't actually care about: "We ran 10,000 scans." "We fixed 500 vulnerabilities." "We have 95% test coverage."

These metrics show effort, not outcomes. Executives hear "we're busy" but get no answers about whether that busyness reduces risk, how risk is trending quarter-over-quarter, or which investments are actually working. When the board asks "What's our risk posture?"—activity metrics can't answer.

💡 What's Needed in 2026

Risk-Based Metrics

Metrics that answer executive questions:

- % of attack surface under test (over time)
- Time to fix critical findings (by risk level)
- Risk posture by business unit
- Unknown → known applications trending
- Risk reduction mapped to investments

This transforms AppSec from cost center to strategic function. Demonstrate risk reduction tied to investment, and you can justify budget. Show risk posture trends, and you prove effectiveness.

🔥 Hot Take

> AI is multiplying risk. Boards are asking harder questions. AppSec teams that can only report activity will lose credibility—and budget. Proving risk is under control is the only answer that matters.
>
> Joni Klippert, Co-Founder and CEO STACKHAWK

How to Achieve It

Proving risk reduction requires infrastructure that tracks the right metrics, processes that turn data into reporting cadences, and people who can translate numbers into narratives.

People

Assign metrics ownership. Someone that ensures accuracy, validates data, reviews reports before distribution, and documents process to prevent drift and maintain credibility.

Train on metrics. The team should understand what each metric measures, how to interpret trends, what actions to take based on changes, and how to explain to stakeholders.

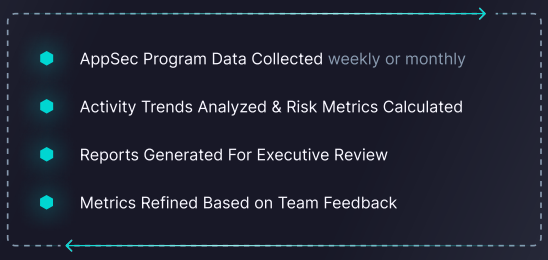
Develop storytelling capability. "Coverage increased from 45% to 75%." "Time to fix decreased by 79%." Train the team to translate metrics into business impact narratives that resonate with executives.

Establish feedback mechanisms. Create channels for executives to share what questions they need answered to ensure your program evolves with the business.

Process

Establish baselines and collection cadence. Document where you started before showing improvement. Track coverage weekly, time-to-fix daily, risk posture monthly, investment ROI quarterly.

Create an executive reporting schedule. Monthly one-pager to executives, quarterly deep dive with board, ad-hoc for critical incidents. Predictability builds credibility.



Technology

Build metrics infrastructure. Automatically measure, report on, and visualize:

- **Attack surface coverage**
  % of discovered assets under active testing
- **Remediation velocity**
  Time from finding to fix; by severity and business unit
- **Risk posture trends**
  Risk score aggregated across applications; over time
- **Investment ROI**
  Risk reduction mapped to total program spend

Enable historical analysis. Store historical data and surface trends. Is risk posture improving? Is remediation velocity increasing? Are coverage gaps closing?

Automate reporting. Scheduled reports, tailored templates for different audiences, export capabilities for GRC workflows.

Build executive-ready visualizations. Communicate risk posture and trends without requiring technical interpretation. Enable drill-down from summary metrics to specific findings.

CONCLUSION

The Intelligence Imperative

Intelligence is the foundation everything else depends on. You can't test effectively without knowing what exists. You can't prioritize without context. You can't prove program effectiveness without measuring risk reduction.

Old Playbook	vs.	New Playbook
⚠ SAST scanning generates thousands of alerts to be triaged		☑ Runtime testing surfaces what's actually exploitable
⚠ Existing application inventorying is reactive or manual		☑ Automated discovery from source code, updated every commit
⚠ Activity metrics (scans run, findings fixed) don't answer business questions		☑ Risk metrics: posture trends, coverage gaps, business impact
⚠ AppSec teams work harder while exposure grows and risks slip through		☑ Intelligence directs effort where it reduces the most risk

About StackHawk

StackHawk reimagines application security for the AI development era. Our AppSec Intelligence Platform combines attack surface discovery from source code with shift-left runtime testing, enabling teams to know what they have, test what matters, and prove risk reduction.

Built for modern development velocity, StackHawk runs natively in CI/CD pipelines to catch exploitable vulnerabilities—including emerging LLM security risks—before they reach production. Organizations use StackHawk to transform AppSec from development bottleneck to competitive enabler.

Learn more at stackhawk.com