Checkmarx | TD SYNNEX Public Sector | DLT | 2025 Report

DevSecOps Evolution:

from DevEx to DevSecOps

Benchmark Your Team's Maturity

Executive Summary

The journey to DevSecOps is in progress, but distance and tough terrain lie ahead. The travellers—DevOps and Security teams—are sharing the road, but they aren't yet in step. They sometimes meet en route, only to diverge again as differing outlooks and priorities prompt a check or swerve.

Yet align they must if they are to achieve the core objective of delivering high-performing code—which we believe must, by definition, be secure code. The situation is growing more pressing because, as our research reveals, modern enterprises have a huge number of development teams and DevOps pipelines.

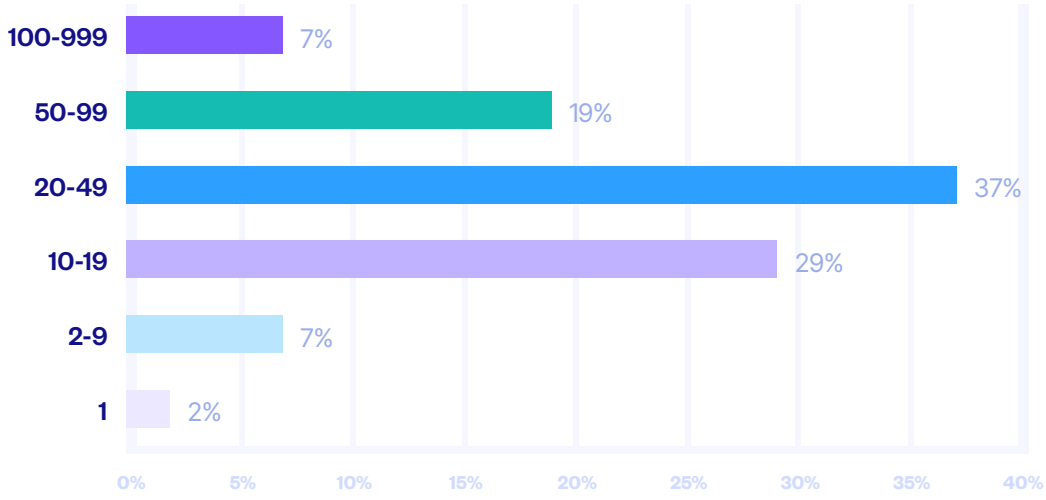


Figure 1: *How many DevOps pipelines does your organization have?*

↘

Our study shows that agile is now the dominant development method used to achieve the required software delivery cadence, but security often fails to match the required pace. This is a problem because fast-in-the-moment, while valuable to a dev meeting their milestones, is certainly not valuable to the business as a whole if missed vulnerabilities lead to a breach that disrupts the business further down the line.

DevEx is a priority

From the security team perspective, developer experience (DevEx) is increasingly a priority. This is a welcome evolution from earlier approaches that largely focused on finding vulnerabilities and getting them in front of developers, without considering the disruptive effect being confronted with a long and often arbitrary list of issues would have on developer flow. Security teams have recognized that their responsibility doesn't end once they "throw the vulnerabilities over the wall"; they need to help developers fix the issues, too. As a result, current focus is on providing tools that integrate into IDEs and don't disrupt the development process. This is a positive step, but it's far from the final destination.

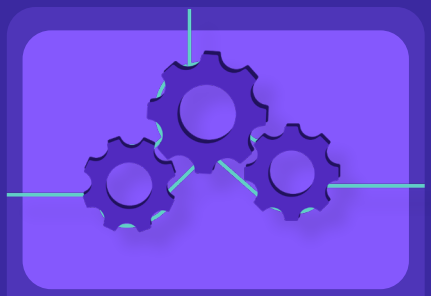
To achieve true DevSecOps, security teams now need to reduce friction at scale to match security to the pace of DevOps. Automation is part of the answer, but it will only work if there is a culture of trust between security and DevOps teams founded on an alignment of goals and ways of working together – from policies and protocols to shared workflows. Once this is established, shared DevSecOps metrics can measure how successfully they are working together. Our research shows that few organizations have achieved this level of maturity.

Security is no longer the gatekeeper for tools

Nevertheless, there are signs that progress is being made. Effective DevSecOps depends on security tools being accepted as core developer tools—not as a bolt-on or supplementary feature. This means developers need to be leading on tool specification and selection and our survey finds that they have stepped into this role. AppSec buying decisions are now increasingly rooted in the development and product teams, with security providing consultation and implementation guidance.

There's evidence of culture change, too. Developers believe that they are getting better at secure coding, and they are generally positive about the training they receive, indicating that they are engaged and seeking to improve. However, they are still spending a disproportionate amount of time on security tasks.

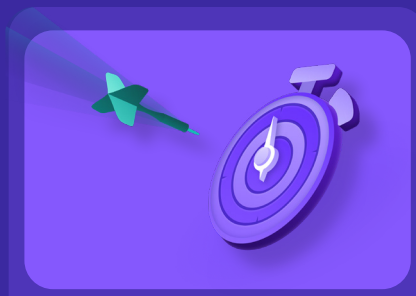
Keep reading for more survey results that show how organizations are tackling the five key requirements for DevSecOps success.



#01

Integrations

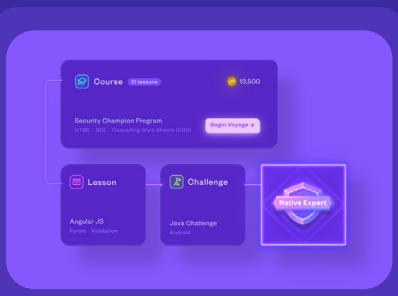
encourage security teams to consider how developers work and how to keep them productive.



#02

Building Shared Measurements

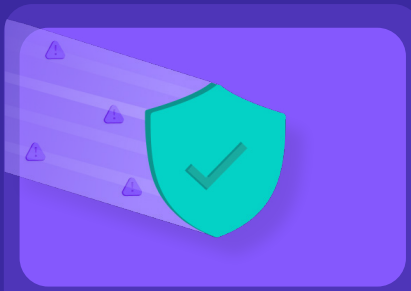
aligns teams on their goals and fosters a shared language and outcomes they can agree on.



#03

Security Education

helps developers understand security, while also enhancing their own skillset advancing their careers, and streamlining security tasks.



#04

Matching Security Velocity to DevOps

helps developers feel like security is part of the process rather than an obstacle.

#05

Automations

are the core of DevOps, fostering efficiency and boosting team satisfaction.

Table of Contents

Introduction & Methodology

Checkmarx undertook this research to learn what DevOps teams think of AppSec and establish where they stand on the road to DevSecOps maturity.

The survey explored the current scale of software development, the methodologies used, how security tasks impact DevOps, and the relationships between security and development teams. It also examined organizations' approach to AppSec, from tool procurement and implementation to measurement metrics, training, and automation.

The research cohort consisted of 1500 Heads of Development, Platform Engineers and Developers/ Software Engineers across North America (USA), Europe (UK, France, Germany, Austria, Switzerland) and APAC (Australia, New Zealand, Singapore). The field research took place in December 2024.

DevSecOps in 2025: Where Do We Really Stand?

A decade since DevOps became mainstream in software development, its prevailing culture prioritizes performance: high-performing teams creating high-performance code.

DevSecOps is the necessary evolution of DevOps in a world where code cannot be termed "high-performing" if it is insecure. The transition to DevSecOps requires a cultural shift that brings together the very different character traits of "move fast and break things" developer teams and "break nothing, protect everything" security teams to achieve the common goal of delivering secure, high-performance code at the pace the business needs.

This isn't something that happens overnight. It takes time to build the understanding and trust needed to foster confidence around security automations and agree on joint success metrics.

The DevSecOps Maturity Model

Typically, organizations are traveling through a maturity model that corresponds with the four stages below:

Stage 0

Reactive Security

- High friction.
- AppSec is bolted onto development, creating a bottleneck and acting as a brake on deployment.
- No relationship between Security and DevOps teams.
- No security training for developers.
- No common performance metrics.

Self-reported DevSecOps maturity status

Stage 1

Security-focused

- AppSec identifies vulnerabilities and throws them over the wall to fix.
- Developers are bombarded with alerts with little or no remediation guidance.
- Relationship between DevOps and Security is adversarial.
- Limited security training for developers. Developers learn through others' experience.
- No common performance metrics.

12%
of organizations

Stage 2

DevEx-focused

- Tools are integrated into the IDE enabling developers to fix identified vulnerabilities without disrupting their workflow.
- Remediation guidance is provided at point of need. Developers don't have to research fixes.
- Formal training is in place.
- Relationship between DevOps and Security teams is positive, but they maintain different performance metrics.

58%
of organizations

Stage 3

Mature DevSecOps

- DevSecOps culture established.
- Security and Development teams agree on joint policies, governance, and methods of collaboration.
- Security automations are designed based on trust between DevOps and Security teams.
- Training is provided formally, at point of need, and in-line within the IDE.
- Goals are aligned and shared performance metrics are monitored.
- High-performance code = secure code

30%
of organizations

Reality Check:
Most Organizations are in Stage 2

Successfully transitioning to DevSecOps needs developers to be ***engaged*** with security but not ***constrained*** by it.

The good news is that developers believe they are getting better at security, and training is having a positive effect. Almost one-quarter (23%) say security is their top consideration when coding, and 92.5% rank the effectiveness of the security training they receive as medium or high. They are also fairly confident that they understand the results provided in a vulnerability ticket and how it manifests in production (fig. 2).

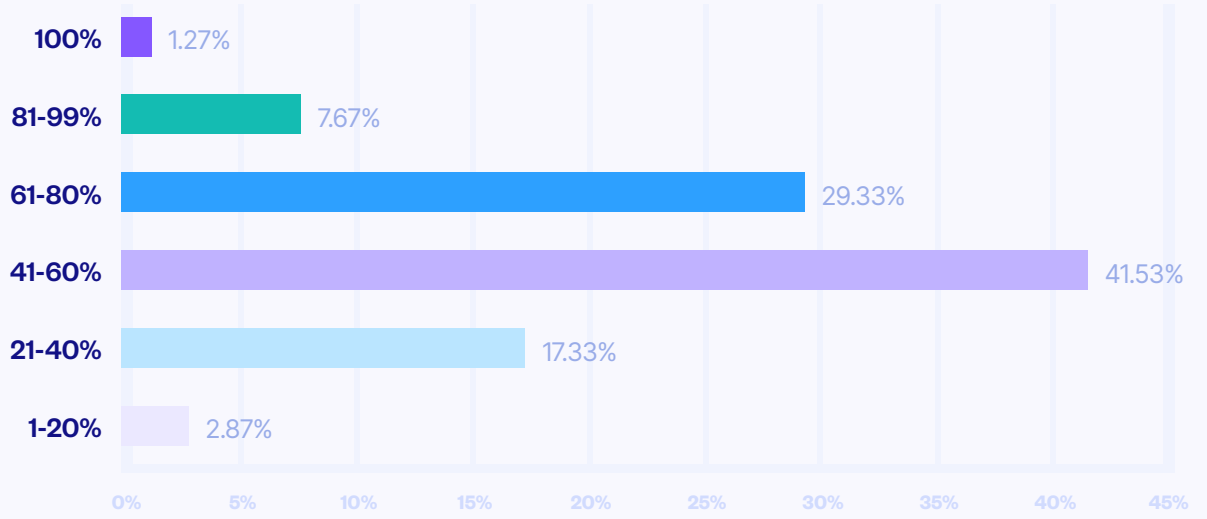
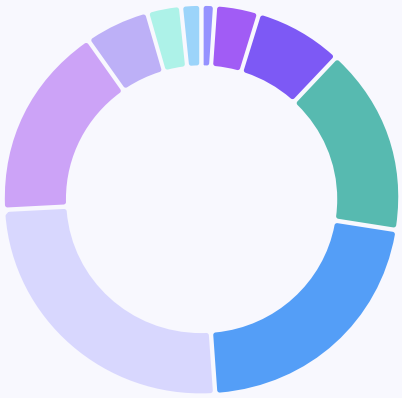


Figure 2: *As a developer, when receiving a vulnerability ticket, on average what percentage, if any, of the time do you understand the results provided and how the vulnerability manifests in production?*

The bad news is that despite feeling like they are doing better at security, they are also spending a considerable amount of time on security. Too much time, many would rightly argue. Seventy-two percent of respondents spend more than 17 hours every week on security-related tasks and one in four spends more than 25 hours. (fig.3). This means security is still significantly intruding on developer flow.



1-4 hours	1.27%	21-24 hours	25.20%
5-8 hours	3.67%	25-28 hours	16.13%
9-12 hours	7.13%	29-34 hours	5.13%
13-16 hours	15.60%	35-36 hours	2.93%
17-20 hours	21.47%	37-40 hours	1.47%

Figure 3: *How many hours each week do you usually spend on security related tasks, if any?*

Security Tool Integrations Point to DevEx Focus

Right now, most security teams are highly focused on developer experience (DevEx). Almost 60% of respondents report that their AppSec and development organizations have agreed to focus on improving the experience of developers via integrations and prioritized results. This is evident in the near-universal integration of tools into IDEs. 99% say at least one tool is integrated, with container and run-time security being the most common integrations, and IaC security less likely to be included.

While DevEx is a vital stepping stone to DevSecOps, it is just that—a stage on the journey. As later analysis shows, far fewer organizations have introduced the collaboration and automation required to take secure software delivery to the next level.

Developer Views on Security: Positive Perceptions But Not Yet a Partnership

As DevSecOps is a culture shift first and foremost, understanding the relationship between Security and DevOps teams is essential.

Most developers believe that AppSec teams are acting with developers' best interests in mind where possible. Asked how in-tune the AppSec team is with release schedules, almost half (47%) feel that they care and are actively working on improving the developer experience with tool integrations, vulnerability prioritization, and remediation guidance. A further 28% say that, although the team cares and attempts to prioritize and give guidance, they don't have the tools and resources to adequately help, and they slow down development as a result.

Figure 4: *What one word best describes the relationship between security and development in your organization, if any?*

The generally positive relationship between development and security teams is further evidenced by the words developers choose to describe it (fig. 4). "Trust" is the most popular choice, with "trustworthy" also appearing, hinting that the cultural evolution fundamental to DevSecOps is in progress. "Risk" and "Policy" are also popular choices, demonstrating an understanding of the rationale for security and its governance needs.

So there's reason to believe that DevOps and Security teams are aligning, but a lack of shared metrics, policies, and governance for 88% of organizations suggests that this is not yet a truly collaborative partnership.

Code-level AppSec is Becoming a Developer-led Discipline

Security tools must be developer tools in effective DevSecOps. From the outset, that means developer teams need to be driving AppSec procurement decision-making. And they are doing just that. Software engineering and product management teams are driving requirements for almost two-thirds of respondents (fig. 5).

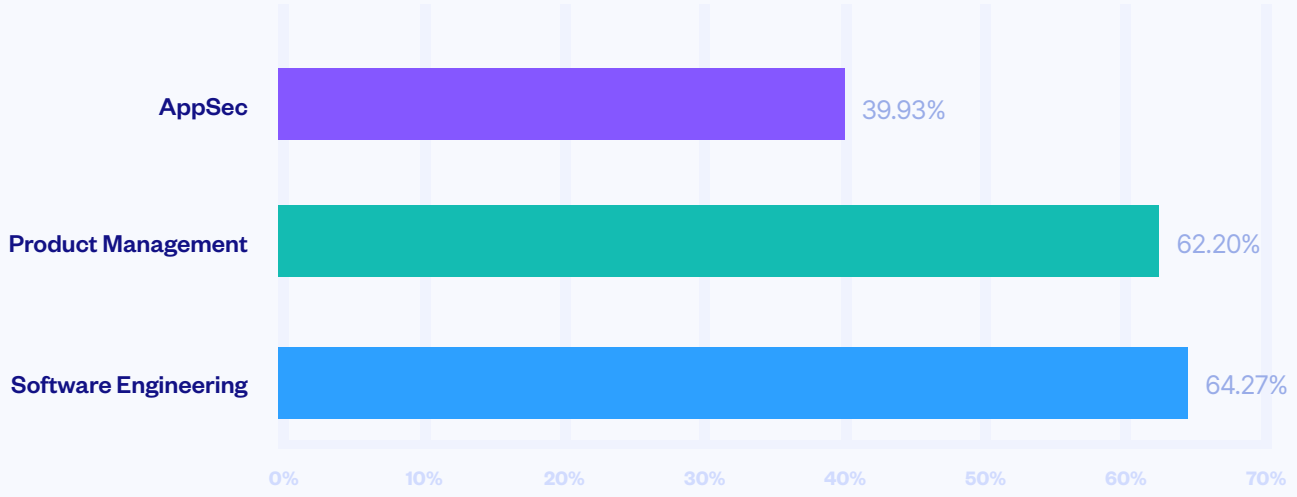


Figure 5: *When buying a security tool, who drives the requirements?*

AppSec teams are playing a supporting role, particularly in implementation, indicating that they accept the mandate for development to select tools, and their own responsibility for ensuring those tools are implemented effectively.

The Pipeline Paradox: No Two DevOps Workflows Look the Same

Standardization is usually a key process design goal, especially when organizations are seeking to progress through a maturity cycle. However, this is a tough challenge when every organization's pipeline setup is different, with potentially multiple pipelines configured differently and variations even within different divisions of the same development organization.

There is no strong consensus on the ways developers receive information on vulnerabilities in need of remediation (fig. 6), implying that each organization uses a mix of methods that have grown organically over time. Relatively fewer respondents report receiving alerts directly in their IDE, while slightly more report receiving direct communications from team leads via email or PDF. This indicates that in many cases vulnerability notifications still tend to break developer flow.

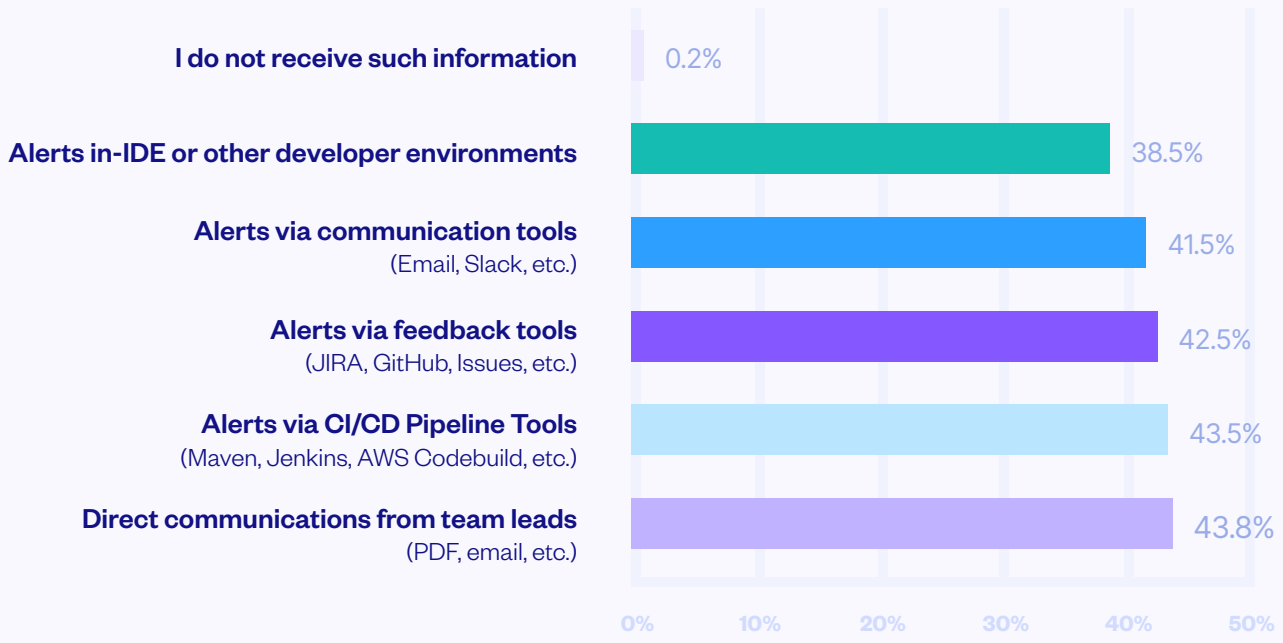


Figure 6: *How, if at all, do you receive information on vulnerabilities in need of remediation? (Select all that apply)*

The fact that AppSec still acts as a brake on software delivery is corroborated by the level of trust in AppSec tools: fifty-five percent of respondents say they have a medium level of trust in the results provided by AppSec tools, but that the rate of false positives is high enough that their velocity is regularly impeded. On the positive side, one-third report high trust in AppSec tools and believe the fixes they make are important.

The perception of why certain security metrics are tracked, and how they are applied to DevOps teams vary according to survey respondents. One-third believe that metrics are tracked to improve product security, while a similar percentage are primarily motivated by compliance demands. A smaller, but still notable, proportion (27%) are more cynical, and believe that their organizations keep metrics simply to justify purchasing the security tool. While this seems like a rather circular activity, it isn't uncommon in organizations where security teams are siloed away from the rest of the business. They know that their work is important but are constantly having to justify investment to stakeholders in other parts of the organization. Consequently, there is little alignment and trust between security and DevOps—the opposite of what is needed to achieve DevSecOps.

Clearly, there is no single way to do DevSecOps, and this is something AppSec vendors must factor into product design and configuration. Flexibility should be favored over enforcing rigid workflows, and tools must align with developer reality, not abstract concepts of best practice.

The right approach is rooted in understanding the existing culture of DevOps and Security teams and identifying what it needs to achieve maturity. This will be different for every business, but there are five key areas where they can focus to move beyond DevEx.

The DevSecOps Maturity Gap: Moving Beyond DevEx

At Checkmarx, we propose five key areas for organizations to target to achieve effective DevSecOps:

01 **Integrations**

02 **Shared measurements**

03 **Security education**

04 **Automation**

05 **Speed**

Earlier in this report we highlighted data showing that organizations are making progress on integrations. Respondents indicated that buying decisions are increasingly developer-led, and security teams are focused on improving DevEx through integrations. From here we'll discuss the other four areas needed to achieve effective DevSecOps.

Shared DevSecOps Measurement

DevSecOps adoption has a lot of room to mature because organizations haven't established meaningful, shared, security metrics that track how quickly and effectively security and developers are working together to solve security issues.

Figure 7 shows DevSecOps metrics and the rate they are currently being tracked. What's interesting here is that there are no standout metrics with near-universal adoption. Contrast this with DevOps, where metrics such as deployment frequency, lead time, and change failure rates are tracked as industry best practice. For DevSecOps to mature on an industry scale, industry standards must be agreed and adopted.

Crucially, most organizations aren't tracking Mean-Time-To-Remediate (MTTR) vulnerabilities and, as developers say they spend many hours every week on security tasks, the likelihood is that they aren't remediating at the speed required. Nor are organizations tracking Security Coverage Per Application (depth of scanning), to ensure that higher risk internet-facing applications get deeper coverage than lower risk internal applications. Instead, they are likely either taking a uniform scanning approach for all applications or have no consistency around scanning practices. Either situation risks wasting time and resources by over-scanning low risk applications or missing important vulnerabilities by under-scanning high priority applications. Overall it indicates that they aren't yet mature enough in DevSecOps to take this kind of strategic approach, which requires deep alignment and trust between DevOps and security teams.

These metrics are critical to DevSecOps because they track whether issues are being remediated at the speed of DevOps – which is essential to achieve the goal of high-performing, secure code.

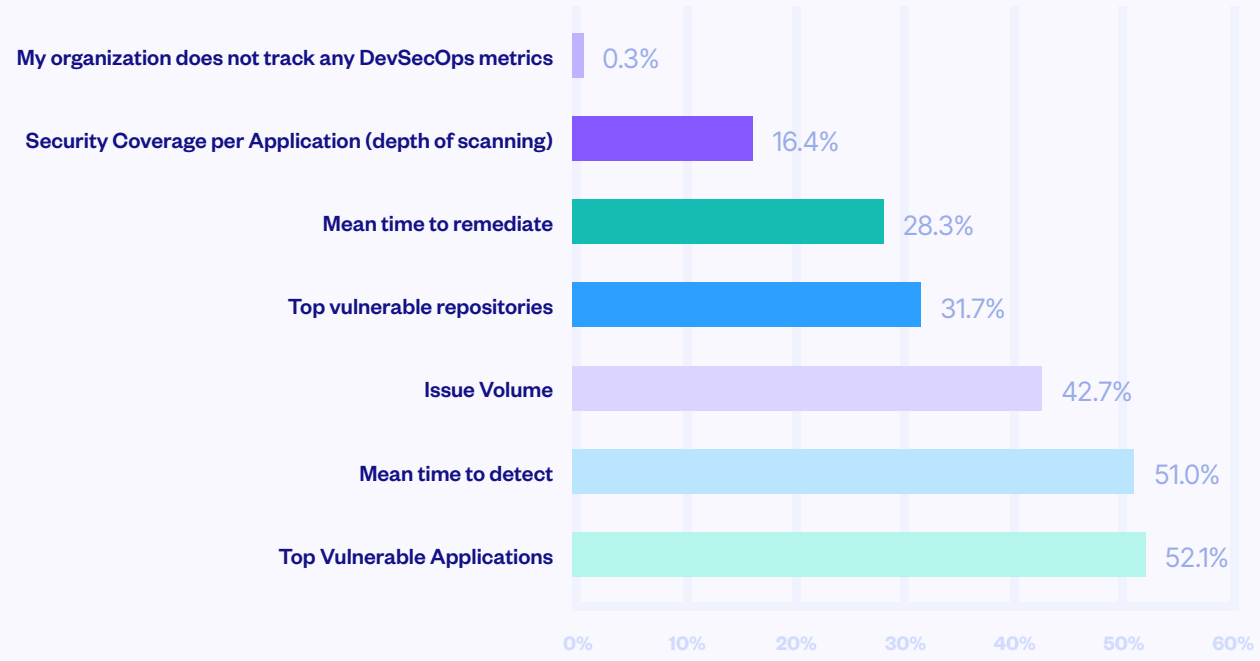


Figure 7: *What DevSecOps metrics does your organization track, if any? (Select all that apply)*

We also found that there is less focus on cycle time in terms of developer metrics (fig. 8), meaning developers are under more pressure to fix bugs by volume than they are by how fast they fix vulnerabilities from the point they are identified. This could create an unintended consequence where developers focus on fixing the low-hanging fruit, regardless of criticality.

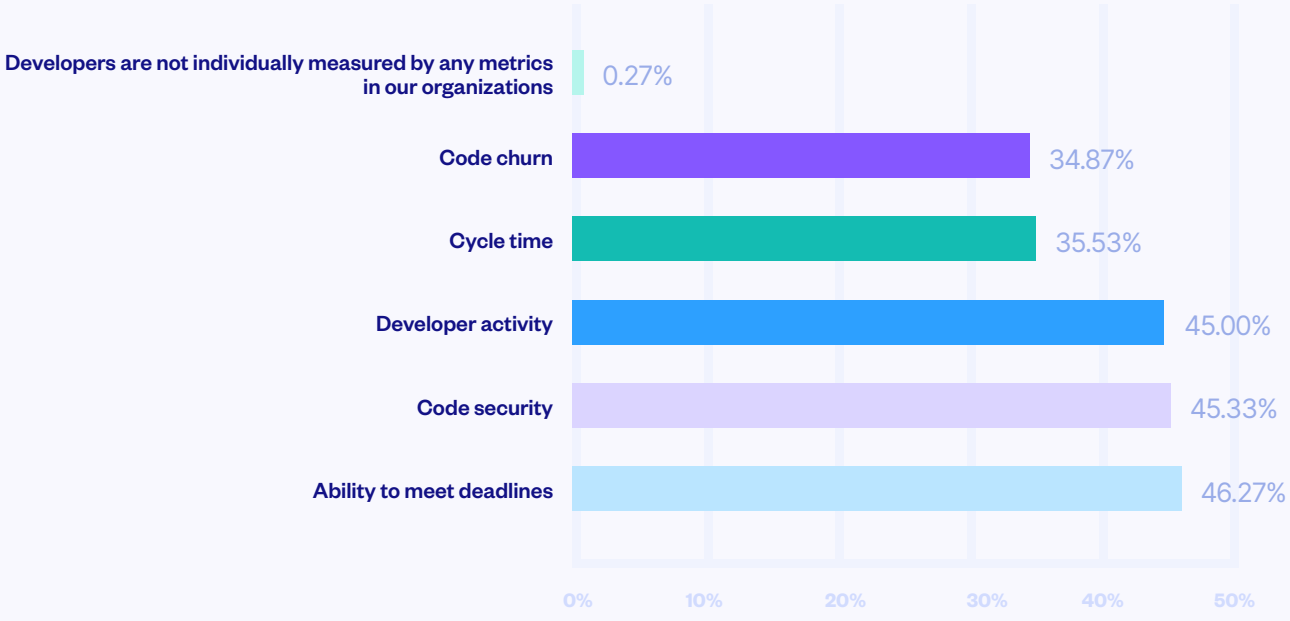


Figure 8: *What metrics are developers individually measured by within your organization, if any? (Select all that apply)*

Security Education Has Come a Long Way... What's Next?

Developers are better trained than ever.

have access to security training or guidance

rate training effectiveness as medium or high

run a security champion program

So, now that security education is no longer a rarity, it's time security teams stopped talking in terms of introducing security training programs and started exploring how to optimize them. To fully realize DevSecOps, we need to work out how to make security training and support more effective so developers can remediate faster.

That means giving developers options to access training in the way that suits them, whenever they need it. While formal training programs have a role to play, the premium on developer time makes investing in standalone training a challenge for many. This reality likely means a pivot towards more just-in-time training, providing remediation guidance attached to vulnerability notifications so developers don't have to break their workflow to research the answer to a problem. Even better is in-the-moment training through in-line feedback within the IDE that allows developers to maintain their flow.

The Automation Challenge: Scaling AppSec Without Breaking DevOps

↘

If you want to reduce friction in AppSec, automation is the way to do it. But it isn't easy, and currently, it isn't really happening on a scale that will shift most teams towards effective DevSecOps.

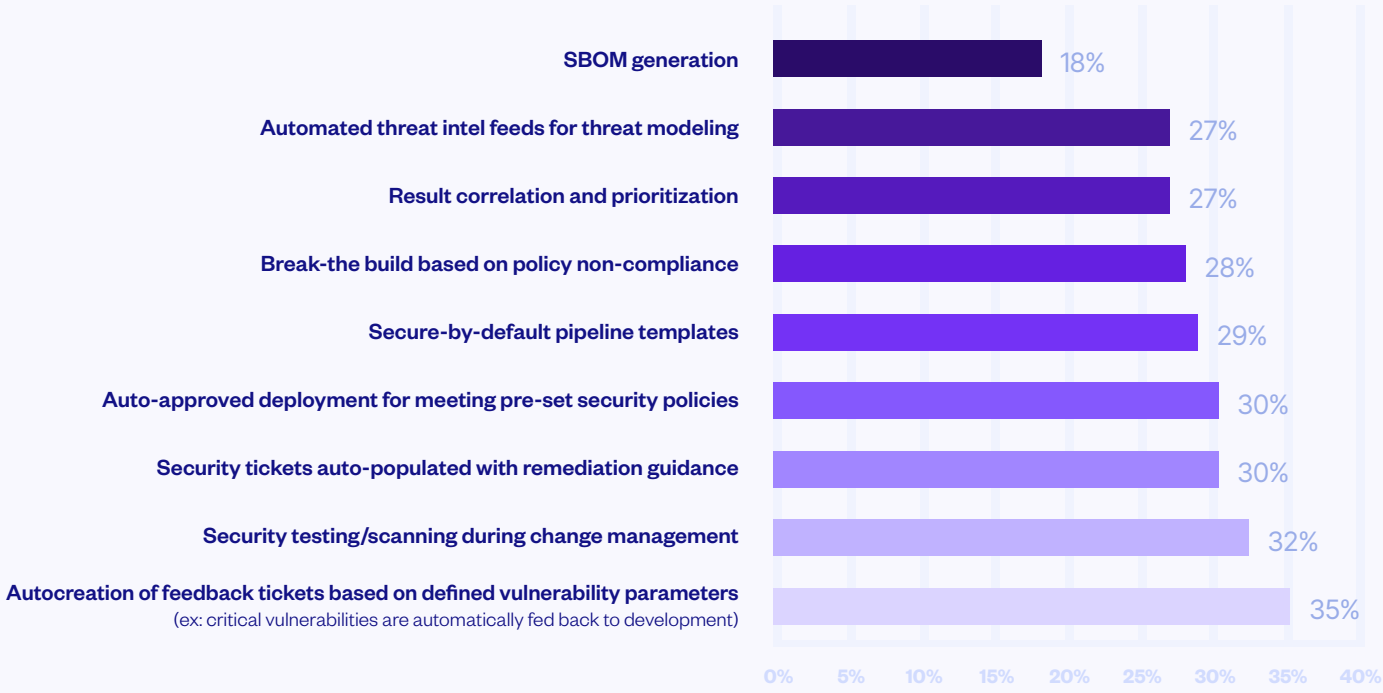


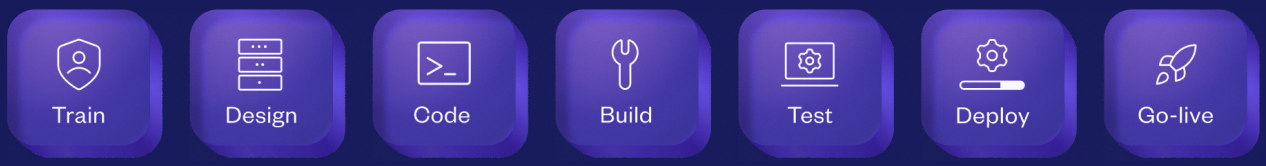
Figure 9: *What AppSec automations does your team currently utilize? (Select all that apply) – organizations that have AppSec automations in place n = 1208*

One in five organizations has no AppSec automation in place at all. Among those that do, only 32% have automated security testing/scanning during change management and only 30% auto-populate security tickets with remediation guidance. Compliance-focused automations such as SBOM generation – arguably of high value to the business but least practical interest to developers – is used by just 18% (fig. 9).

The low reported adoption of key AppSec automations is a further finding that highlights the current lack of industry standards in this area. Additionally, as automation is a core goal of DevOps, the fact it isn't being achieved for security means we are not reaching DevSecOps.

What Security Controls Can Be Automated?

Where to look first for efficiency



- Secure code training assignment
- Security tickets auto-populated with remediation guidance

- Secure-by-default pipeline templates
- Automated threat intel feeds

- In-line scanning
- Security testing auto-initiation during change management
- Compliance automation can break the build
- Auto-creation of feedback tickets based on defined parameters

- SBOM generation
- Results correlation & prioritization
- Fast lanes for policy compliance

Why "Automate Everything" Will Fail in DevSecOps

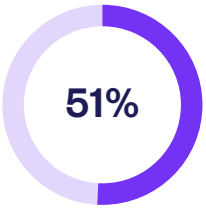
Automation is one of the most difficult aspects of DevSecOps to achieve, because it relies on the other key elements—and in particular DevSecOps culture—being already in place.

Organizations that simply try to "automate everything" will fail because, without joint policies, shared measurements, mutually agreed governance, and trust between Security and DevOps, there is no foundation on which to design automations that will work and be accepted by both teams. Currently, because every organization's pipelines setup looks different and there's no consistency over DevSecOps metrics, every organization's automations need to be different too. There is no one-size-fits-all.

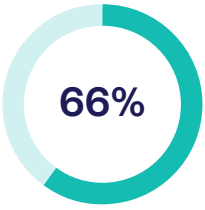
In this highly diverse environment, scaling DevSecOps becomes a big challenge. It's relatively simple to implement security automations for small teams, but faced with the hundreds of DevOps pipelines reported by our survey participants, organizations need to take a strategic approach that starts with culture and collaboration and proceeds to the agreement on industry standards for metrics and automations. Only by doing this will we get the consistency needed for organizations to achieve Stage 3 of DevSecOps.

Speed – DevSecOps Speed is Not the Same as Tool Speed

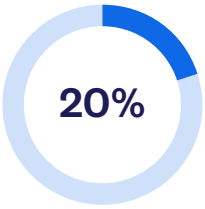
The goal of DevSecOps is to deliver high-performance secure code fast. But, too often, teams equate speed with how fast tools can scan and deliver results integrated into the IDE. Yet this is just a more sophisticated and faster way of "throwing results over the wall". And it really isn't working, because:



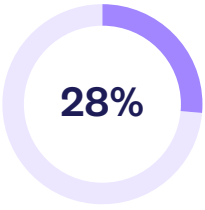
of developers spend more than 21 hours per week on security tasks.



say that AppSec tool false positive rates are high enough that their velocity is regularly impeded.



of developers understands the information they receive in a vulnerability ticket less than 40% of the time, meaning they need to spend time researching fixes.



of developers say their AppSec team doesn't have the tools or resources to help them fix security issues, which slows down development.

The fastest scans in the world are no good if they hit a brick wall as soon as they get to the remediation stage.

Developers must be empowered to code securely by design—through policy-driven secure-by-default pipeline templates, for example—and remediate fast with assistance such as just-in-time training, trusted scanning, and GenAI remediation guidance. This means they spend less time on security tasks as a proportion of their workday. Integrations, shared measurements and policies, dynamic security education, and automation are all foundational to achieving this outcome.

Recommendations

The cultural shift needed to achieve DevSecOps means that DevOps and Security teams need to be traveling the road in step, surmounting cultural differences to find common ground, where neither speed nor security is compromised, and high-performing code is the result.

Focus on Developer Experience Through Collaboration

For security and development leaders, this means continuing with the focus on developer experience, but taking it to the next level, fostering collaboration to build foundations for policy and governance on which trusted automations can be built.

Foster Developer Advocacy

Developers can support this agenda by highlighting the security component of their work schedule and pushing for solutions that help them address security more efficiently.

Track Qualitative DevSecOps Metrics

Leaders should normalize tracking DevSecOps metrics that go beyond measuring the extent of security tool adoption, instead monitoring security effectiveness in more qualitative terms based on how well security and DevOps are working together.

Cultivate a Shared Culture

Leaders should also encourage a shared culture in which DevOps and security teams work to actively understand and support each other's way of working.

Conclusion and Future Outlook

Our research shows organizations are well on the road to DevSecOps, but they need to understand that the next era of DevSecOps isn't just about improving DevEx—although that's still important—it's about integrating security at scale without compromising speed.

Predictions for DevSecOps in 2026 and beyond:

Organizations will start to focus on scaling DevSecOps and, as they do so, we will start to see greater consensus emerge around the metrics and automations that constitute DevSecOps (not just DevOps) best practice.

These nascent industry standards will provide direction for organizations seeking to increase DevSecOps maturity, but we'll still see attempts to "automate everything" in a bid to run before they can walk. If organizations don't succeed in establishing trust and shared governance between DevOps and Security teams, this approach will fail.

Thoughtful organizations will work on promoting collaboration and encouraging Security and DevOps teams to develop joint policies and governance, incrementally improving DevSecOps efficiency and achieving the culture shift needed to sustain it as the best practice for modern software engineering.