BLACKDUCK®

Global State of DevSecOps 2024

Insights and Trends in Software Security Testing from Black Duck

Table of Contents

Table of Contents

Executive Summary

It is a time of radical change in software development, with organizations in every industry recognizing the need for robust, efficient security processes that can keep pace with new development practices, such as AI-assisted coding.

The findings in the “Global State of DevSecOps 2024” report are based on a comprehensive survey that Black Duck® commissioned from Censuswide, an international market research consultancy. More than 1,000 software developers, application security (AppSec) professionals, CISOs, and DevOps engineers across multiple countries and industries were included in the survey.

This report provides critical insights into the current state of DevSecOps practices and AppSec testing. It delivers a comprehensive analysis of trends, challenges, and opportunities, and it offers actionable insights for organizations seeking to enhance their DevSecOps practices.

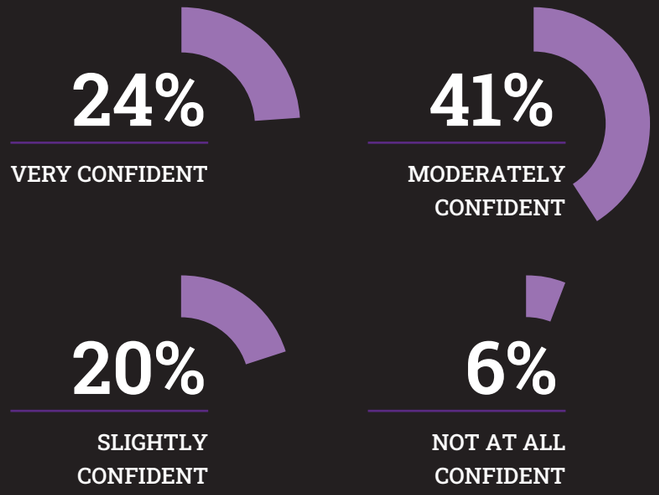
About Black Duck

Formerly the Synopsys Software Integrity Group, Black Duck offers the most comprehensive, powerful, and trusted portfolio of AppSec solutions in the industry. We have an unmatched track record of helping organizations secure their software quickly, integrate security efficiently in their development environments, and safely innovate with new technologies.

Findings Overview

Although 85% of respondents to our survey say they have some measures in place to address the challenges posed by AI-generated code, less than a quarter were very confident in their policies and processes for testing such code.

Here is the breakdown.



AI-assisted development soars but securing AI-generated code lags far behind

One of the most striking discoveries in this report is that the AI revolution is already over—and AI won, at least when it comes to integrating AI into software development processes. The adoption of AI in software development has gone beyond a tipping point, with over 90% of the respondents to our survey using AI assistance in some capacity.

Parallels between securing AI-generated code and securing open source

The rapid adoption of AI-assisted coding by software development teams shares several similarities with the historic rise of open source software use. Both movements disrupted traditional software development practices. Open source challenged proprietary software models, and AI-assisted coding is transforming how code is written and reviewed.

But just as with open source use, bringing AI-assisted coding tools into software development presents unique intellectual property (IP), licensing, and security challenges that need careful management by development teams. For example, both unmanaged open source and AI-generated code can create ambiguity about IP ownership and licensing—especially when the AI model uses datasets that might include open source or other third-party code without attribution.

AI-assisted coding tools also have the potential to introduce security vulnerabilities into codebases. One researcher flatly concludes that "autogenerated code cannot be blindly trusted, and still requires a security review to avoid introducing software vulnerabilities."

There are clear challenges in managing and securing AI-generated code. Our survey found that organizations are at different stages of implementing policies and controls around AI tool usage, reflecting the nascent nature of this trend.

Although 85% of respondents to our survey say they have some measures in place to address the challenges posed by AI-generated code, only 24% are "very confident" in their policies and processes for testing such code. A total 67% of respondents feel only "moderately confident" (41%), "slightly confident" (20%), or "not at all confident" (6%).

This lack of confidence may reflect the fact that that 21% of respondents acknowledge that their development teams are bypassing corporate policies and using unsanctioned—and, one would assume, unsupervised—AI tools. Again, unmanaged AI use parallels the early days of unmanaged open source use, when few executives were aware that their development teams were incorporating open source libraries into proprietary code, let alone the extent of that use.

An increased focus on software security testing

Test coverage is substantial but not universal, with 57% of respondents testing between 41% to 80% of their projects, branches, and repositories, suggesting opportunities for expanding security test coverage.

Our findings show that organizations are prioritizing security testing based on the sensitivity of information handled (37% of respondents), while also emphasizing industry best practices (36%) and increasing use of automated security testing (35%).

Configuration of security tests is becoming more centralized, with 55% of respondents using centralized interfaces for test configuration. And although their execution is becoming more automated, the persistence of nonautomated activities documented in this report indicates substantial

Findings Overview

room for improvement. A significant percentage of respondents still uses manual processes in their application security testing and remediation workflows. The exact amount varies depending on which manual process we look at, but it ranges from about 15% to 43% of respondents.

Too much noise, too many tools

A slight majority of respondents find security test results at least "somewhat easy" (52%) to understand and act upon, while another 20% deem their results "extremely easy" to understand. However, this perception varies across roles, industries, and geographies.

The findings also reveal a critical challenge with "noise" in security testing results; that is, output that is considered irrelevant or not worth acting upon. Noise is often caused by a high number of false positives or a large volume of duplicative true positives in results. Sixty percent of respondents reported that they consider over 20% of their results as noise, impacting efficiency and decision-making processes.

Despite a broader trend of integrating security into development processes, 61% of respondents report that security testing moderately or severely slows down development. The tension between security and development speed remains a critical challenge for every industry.

The fact that 82% of organizations use between 6 and 20 security testing tools is certainly a factor, with a broad proliferation of tools contributing to the high levels of noise reported by respondents. Multiple tools may detect the same issues, leading to duplicative results. Or different tools may provide conflicting results for the same code or application. Each tool may generate its own false positives, which compounds as more tools are used.

With so many tools in use, organizations are struggling to effectively integrate and correlate results across platforms and pipelines, leading to difficulty distinguishing between genuine issues and false positives, as well as challenges in prioritizing issues across different tools' outputs.

Looking ahead

Several key trends are shaping the path of DevSecOps.

- **Increased automation of security testing and remediation processes**
- **A need for policies concerning the use of AI-assisted development tools**
- **Enhanced focus on reducing noise in security test results to improve efficiency**
- **The evolution of cross-functional collaboration in security decision-making**

Organizations have significant opportunities to improve their DevSecOps practices by leveraging automation, enhancing the clarity of security test results, developing robust policies for AI-assisted development, and fostering better cross-functional collaboration.

As the landscape continues to evolve, organizations must stay agile, adapting their AppSec processes to meet emerging challenges. The most successful will be those that can effectively balance rigorous security practices with the speed and innovation demands of modern software development.

A Deep Dive into the State of DevSecOps in 2024

Our survey of over 1,000 security professionals reveals a state of flux, with organizations striving to balance security measures with the demands of rapid development cycles. This section delves into the current state of AppSec testing and highlights key trends, challenges, and opportunities that define the testing landscape in 2024.

Q1. Which of the following criteria does your organization consider when determining which application security tests to run and when they are run?

Sensitivity of information accessed/transmitted by the application	37%
General best practices recommended by third-party organizations (e.g., OWASP)	36%
Ease-of-configuration or automation of the security tests	35%

Three priorities are driving security testing

Our results reveal that respondents to our survey have a clear set of priorities for effective security testing. Protecting sensitive information is a key mandate for security teams. Development teams value efficiency through automation and closed feedback loops, and implementing best practices for resilient pipelines is fundamental to operations teams.

Protecting sensitive information

The foremost consideration, cited by 37% of respondents, is protecting the sensitive information accessed or transmitted by the application. Taking a risk-based approach as these organizations are doing reflects a mature understanding of the impact potential breaches can have across different parts of an application ecosystem.

In a recent analysis of 1,300 customer applications, Black Duck found sensitive data exposure issues affecting 86% of those customers, accounting for over 30,000 vulnerabilities, including 4,800 critical-risk instances. Sensitive data exposure is one of the most common and serious security issues across industries. To address these vulnerabilities, organizations need to implement strong encryption practices, use up-to-date security protocols, and ensure that sensitive data is properly protected both when it's being transmitted and when it's stored.

Our data shows that organizations in sectors such as Application/Software, Banking/Finance, Healthcare, and Government are particularly attuned to this priority, given the highly sensitive nature of the data they handle.

Adhering to best practices

Thirty-six percent of organizations rely on the best practices recommended by third-party organizations like OWASP. Adherence to established guidelines ensures a baseline of security across diverse development environments. However, it also raises questions about the adaptability of these standards in the face of rapidly evolving threats.

Industry standards may have difficulty adapting in the face of rapidly evolving threats. For example, OWASP standards have yet to address the unique security challenges posed by AI-generated code.

Automating and ensuring ease of test configuration

The emphasis on automation and ease of test configuration, prioritized by 35% of respondents, underscores the growing integration of security into DevOps processes. This move toward DevSecOps reflects the recognition that security must be woven into the fabric of the development life cycle rather than treating it as an afterthought.

Trending toward centralization

Q2. Which statement best describes your process of configuring and running application security tests across your SDLC or CI pipeline?

Testing tools provided by the same vendor are configured using a centralized interface and automatically run with policies	30%
All tests are configured using a centralized interface and automatically run with policies	26%

The top responses to the survey's Question 2 reveal a clear trend toward centralization in tool configuration for efficiency and consistency. Thirty percent of respondents reported using a vendor's interface to configure tests from that vendor, while 26% reported using a centralized interface for all tests, regardless of vendor.

Centralizing security tools allows for a unified management interface, which simplifies the monitoring and configuration of security measures. This reduces the complexity associated with managing multiple disparate systems, facilitates integration at each stage of the pipeline, and ensures that security policies are consistently applied across the organization. With a centralized system, security efforts can be more easily coordinated, reducing the likelihood of gaps or overlaps in security coverage. A centralized, holistic approach enhances the ability to detect and respond to threats across the entire IT infrastructure.

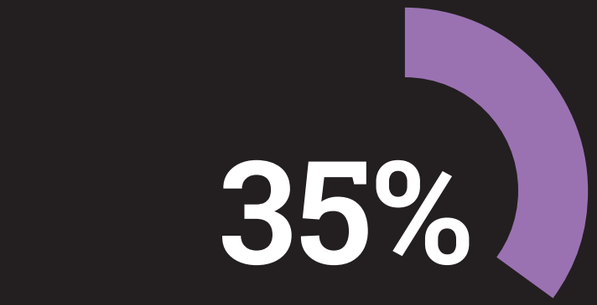
Centralized management also allows better visibility into an application's security profile, enabling more effective identification and mitigation of vulnerabilities. Further, it facilitates the collection and analysis of security data, which is crucial for proactive threat detection and response.

Overall, centralization and vendor consolidation in security testing can significantly enhance an organization's ability to protect its digital assets by simplifying management, improving coordination, and potentially reducing costs.

A struggle to attain full security coverage

Q3. Which of the following statements best describes the manner in which new projects, branches, or repositories are added to your application security testing queue?

All are added to the test queue manually (e.g., declared by dev team, selected by security team)	29%
All are added to the test queue automatically (e.g., detected by testing tools)	38%
Most are added to the test queue automatically; a few are added manually	22%
Most are added to the test queue manually; a few are added automatically	6%
I am not familiar with how items are added to the security testing queue	4%



About 35% of organizations are still heavily reliant on manual intervention in their security testing queue management.

Q4. Approximately what percentage of your projects, branches, and repositories are included in your application security testing queue?

Percentage of projects, branches, and repositories included in testing queue	Percentage of respondents
41%–60%	37%
61%–80%	21%

Despite the emphasis on comprehensive security, many organizations struggle to achieve full coverage, as the responses to Questions 3 and 4 demonstrate. Nearly 30% of respondents still add new projects, branches, or repositories to their application security testing queue manually. Six percent use mostly manual processes with some automation. In other words, about 35% of organizations are still heavily reliant on manual intervention in their security testing queue management.

While there are varying perceptions of the extent to which security testing impacts development workflows, survey results show a clear correlation between the perceived impact on testing and manual processes. For example, 50% of those that say application security testing slows down the process also say that most projects are added to the test queue manually.

However, 38% of respondents report that they are taking full advantage of automated processes to include all projects in test queues, and another 22% report mostly using automated processes. This means that 60% of organizations are leveraging automation to a significant degree in their security testing workflows.

Thirty-seven percent of respondents include only 41% to 60% of their projects, branches, and repositories in their testing queue. Twenty-one percent achieve 61% to 80% coverage.

This coverage gap presents significant risk, potentially leaving critical parts of an organization's application ecosystem untested. While counterintuitive, some respondents noted slightly higher-than-average coverage despite using manual processes to add projects to the test queue. This may simply be the level of coverage being perceived as higher due to the greater level of effort to test each project.

Who determines when security tests are run

Q5. Which of the following teams/departments determine which application security tests are performed, when, and on which projects?

Security	44%
Development/software engineering	42%
DevOps	37%
Quality assurance	34%
Compliance	28%
Cross-functional groups	21%
Legal	19%
None of the above	1%

The responses to Question 5 offer valuable insights into how organizations are structuring their application security testing decisions. This data paints a picture of organizations increasingly treating security as a shared responsibility, integrated into various stages of the software development life cycle.

The close percentages for security (44%) and development/ software engineering (42%) suggest a trend toward shared responsibility for security testing. This aligns well with DevSecOps principles, indicating that security is becoming more integrated into the development process.

At 37%, DevOps teams play a significant role in security testing decisions. This further supports the trend toward integrating security throughout the development life cycle. At 34%, QA teams are also heavily involved,

82%

of organizations use between 6 and 20 security testing tools.

suggesting that many organizations view security as an integral part of overall software quality.

The involvement of compliance (28%) and legal (19%) teams indicates that regulatory and legal requirements are significant factors in security testing decisions for many organizations.

Twenty-one percent of respondents indicate that cross-functional groups are involved in these decisions, showing a trend toward collaborative, multidisciplinary approaches to security. With only 1% selecting "None of the above," it's clear that the majority of organizations have specific teams or processes in place for determining security testing.

The distribution across teams suggests a relatively mature approach to security in many organizations, moving away from security as solely the responsibility of a dedicated security team. These results align with broader industry trends toward DevSecOps and "shift-everywhere" security practices, as described in the "Building Security in Maturity Model" report, where security is integrated earlier and more continuously in the development process.

A tool proliferation challenge

Q6. Approximately how many application security testing tools does your organization use?

Number of security testing tools	Percentage of respondents
6–10	34%
11–15	33%
16–20	15%
Total	82%

One of the most striking findings from our survey is the sheer number of security testing tools in use, as shown by the responses to Question 6. Eighty-two percent of organizations use between 6 and 20 security testing tools.

A proliferation of tools, although intended to provide comprehensive coverage, introduces significant complexity in integration, results interpretation, and overall management. It correlates strongly with another key challenge—noise in security testing results.

The noise factor

Q9. Approximately what percentage of security test results are noise? For example: duplicative results, false positives, conflicting with other tests/tools.

Percentage of noise in findings	Percentage of respondents
21%–40%	30%
41%–60%	30%
Total	60%

Question 9 uncovers a significant hurdle in effective security testing: the high level of noise in results. A total of 60% of respondents reported that between 21% and 60% of their security test results are noise. A high noise level can significantly impact the effectiveness of security efforts and lead to efficiency loss, as teams must spend time filtering out irrelevant findings. It can also lead to alert fatigue and genuine threats being overlooked, as well as resource misallocation due to organizations directing too much of their security efforts toward noncritical issues.

Role-based differences

There is a perception among security personnel of a high percentage of noise within security test results. This is likely because security teams are commonly tasked with managing security tests, as they sit toward the top of the review funnel. These teams present dev/engineering teams with cleansed and prioritized results, which in turn results in those teams skewing toward lower perceived noise.

Likewise, 17% of dev/engineering personnel feel they don't have enough visibility into security tests to identify noise in results. This is in stark

contrast to CISOs, CTOs/CPOs, and AppSec professionals; only 1% of respondents in those roles cite a lack of visibility when detecting noisy results. One core tenet of efficient DevSecOps is adequate visibility into software artifacts and associated risks across all teams. Inadequate visibility can slow down issue detection, prioritization, and remediation, and leave pipelines prone to breakdowns and software open to attack.

The AI revolution in security testing

Q14. Are your developers using AI, generative, or transformational tools to write code and modify projects?

Yes (Net)	91%
Yes, all developers are permitted to, and do, use these tools	27%
Yes, but only certain developers/teams are permitted to, and do, use these tools	43%
Yes, while we do not allow the use of these tools, we are aware that some developers use them	21%

Over 90% of organizations are using AI tools in some capacity for software development. The distribution of responses to Question 14 illustrates a seemingly phased adoption curve. Twenty-seven percent of respondents note that all developers are permitted to use AI, generative, or transformational tools in their work, while 43% permit only certain developers or teams to use such tools, and 21% forbid their use alongside an awareness that such tools are, in fact, being used by their developers.

Worldwide AI adoption

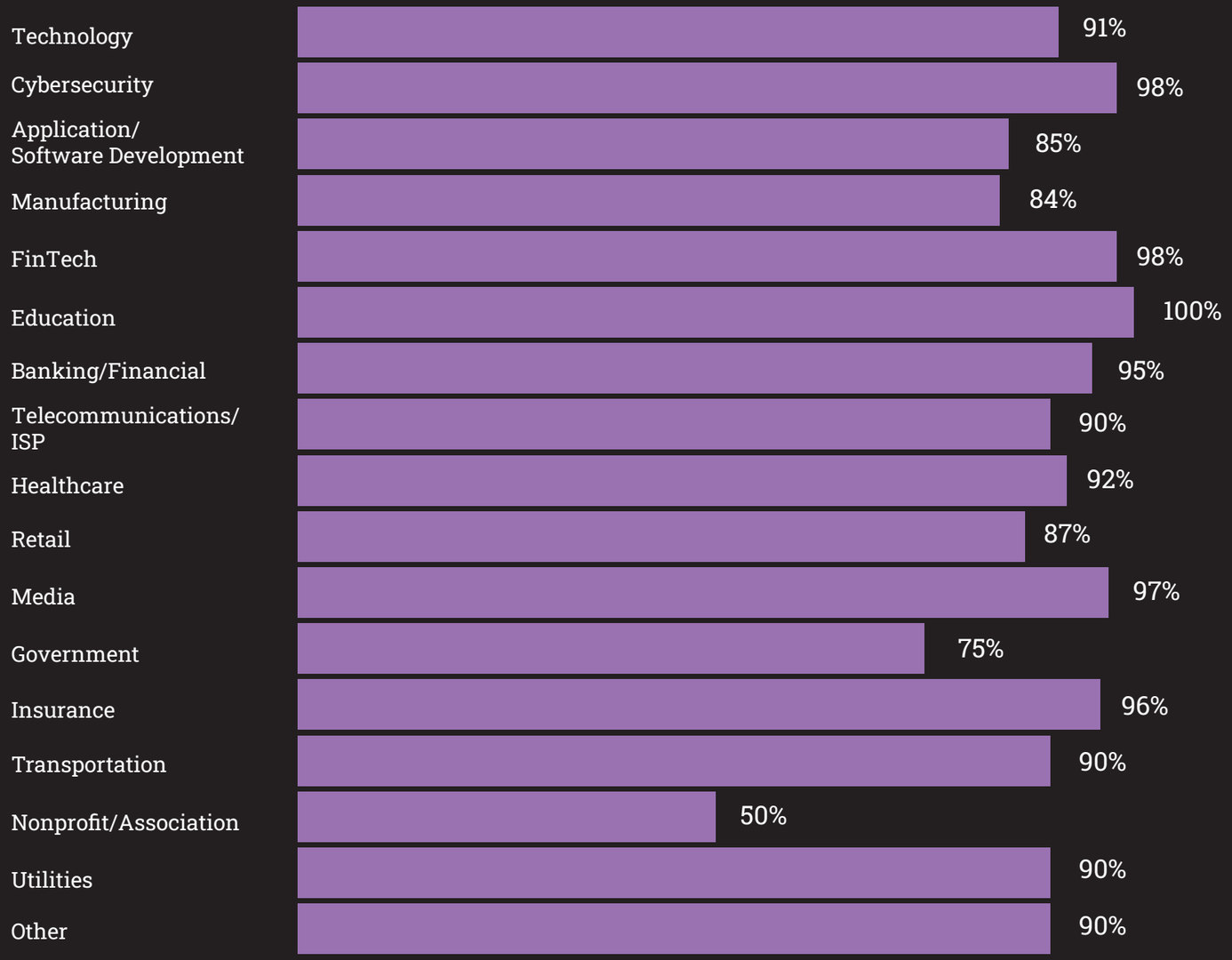
Q14. Are your developers using AI, generative, or transformational tools to write code and modify projects (by region)?

	U.K.	U.S.	France	Germany	Finland	China	Singapore	Japan
Yes (Net)	94%	97%	92%	94%	93%	97%	96%	60%
Yes, all developers are permitted to, and do, use these tools	30%	26%	26%	20%	27%	35%	37%	26%
Yes, but only certain developers/ teams are permitted to, and do, use these tools	39%	46%	41%	50%	51%	50%	37%	45%
Yes, while we do not allow the use of these tools, we are aware that some developers use them	26%	26%	25%	24%	14%	12%	22%	25%
No, developers are not permitted to, and do not, use these tools	4%	2%	6%	6%	5%	3%	2%	3%
I do not have enough visibility into development processes to know if these tools are used	2%	2%	2%	1%	2%	0%	2%	1%

The regional responses to Question 14 demonstrate that AI adoption in software development is not only a phenomenon—it is a global phenomenon, with slight variations in results probably reflecting differences in technological infrastructure, regulatory environments, or cultural attitudes toward AI.

Developers Using AI

Q14 Are your developers using AI, generative, or transformational tools to write code and modify projects (by industry sector)?



Similar numbers play out by industry sector, with over 90% adoption reported across the Technology, Cybersecurity, FinTech, Education, Banking/Financial, Healthcare, Media, Insurance, Transportation, and Utilities sectors. Even lagging sectors, such as Nonprofit, report at least 50% adoption. Perhaps unsurprisingly, the larger the organization, the more likely it has significantly adopted some facet of AI in its software development.

This trend is reshaping the security testing landscape and also introduces new challenges, particularly in securing AI-generated code and managing potential biases or vulnerabilities that AI systems might introduce, as the responses to Question 15 show.

Q15. How confident are you that you have the processes in place to manage and secure AI-generated code?

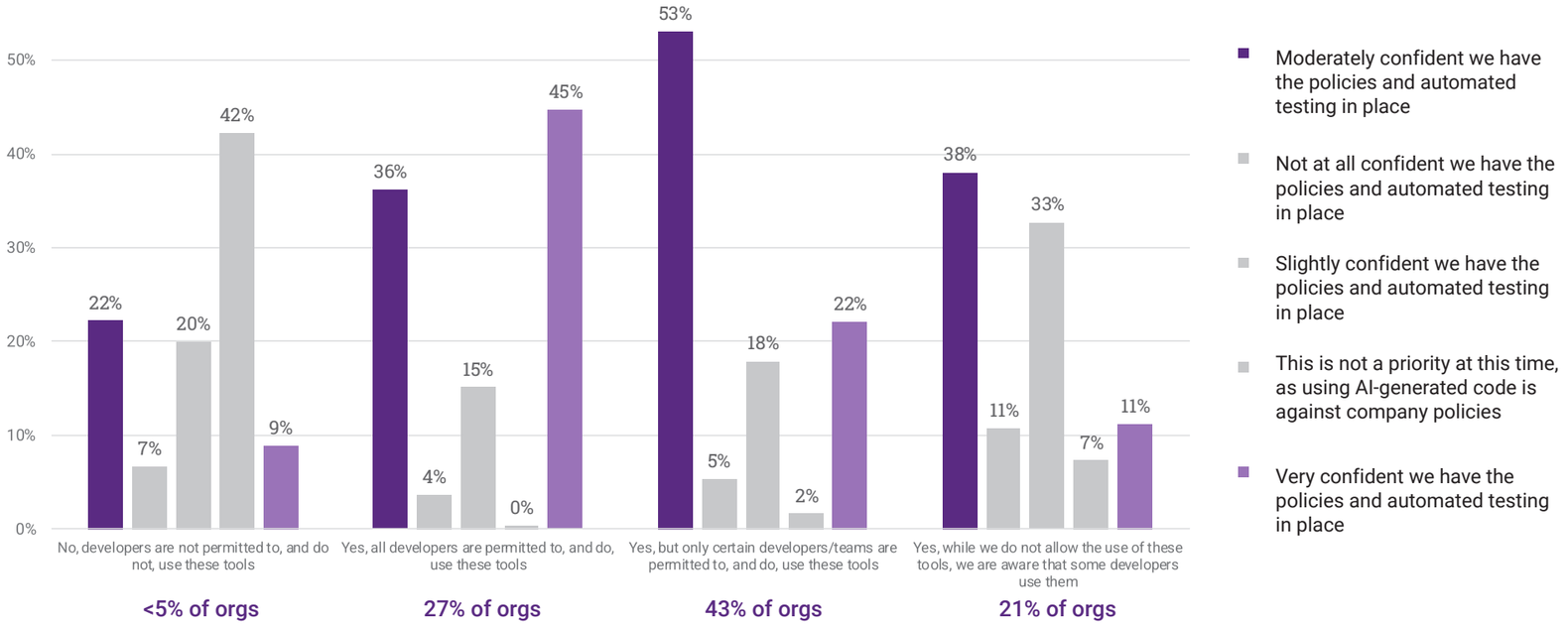
Confident (Net)	85%
Very confident we have the policies and automated testing in place	24%
Moderately confident we have the policies and automated testing in place	41%
Slightly confident we have the policies and automated testing in place	20%
Not at all confident we have the policies and automated testing in place	6%
This is not a priority at this time, as using AI-generated code is against company policies	4%
I do not have enough visibility into our processes to manage and secure AI-generated code	5%

Most respondents not confident they're securing AI-generated code

While the net confidence level of respondents to Question 15 may seem high at first blush, a deeper dive into the responses show that 41% of respondents are only moderately confident that they have the policies and automated testing in place to adequately vet AI-generated code, while 20% are only slightly confident and 6% are not at all confident—a total 67% of respondents altogether showing concern about managing and securing AI-generated code.

This distribution suggests that even though their development teams are adopting AI tools, many organizations are still in the process of putting policies and tools into place to manage the unique challenges posed by AI-generated code. Ensuring the reliability and security of that code remains a significant challenge. As one example, AI tools trained on public open source codebases could introduce potential IP, copyright, and license issues into the code they produce, particularly if that code is used in proprietary software.

Figure 1. Developers' AI usage (permitted or not) correlated against moderate to high confidence in security controls



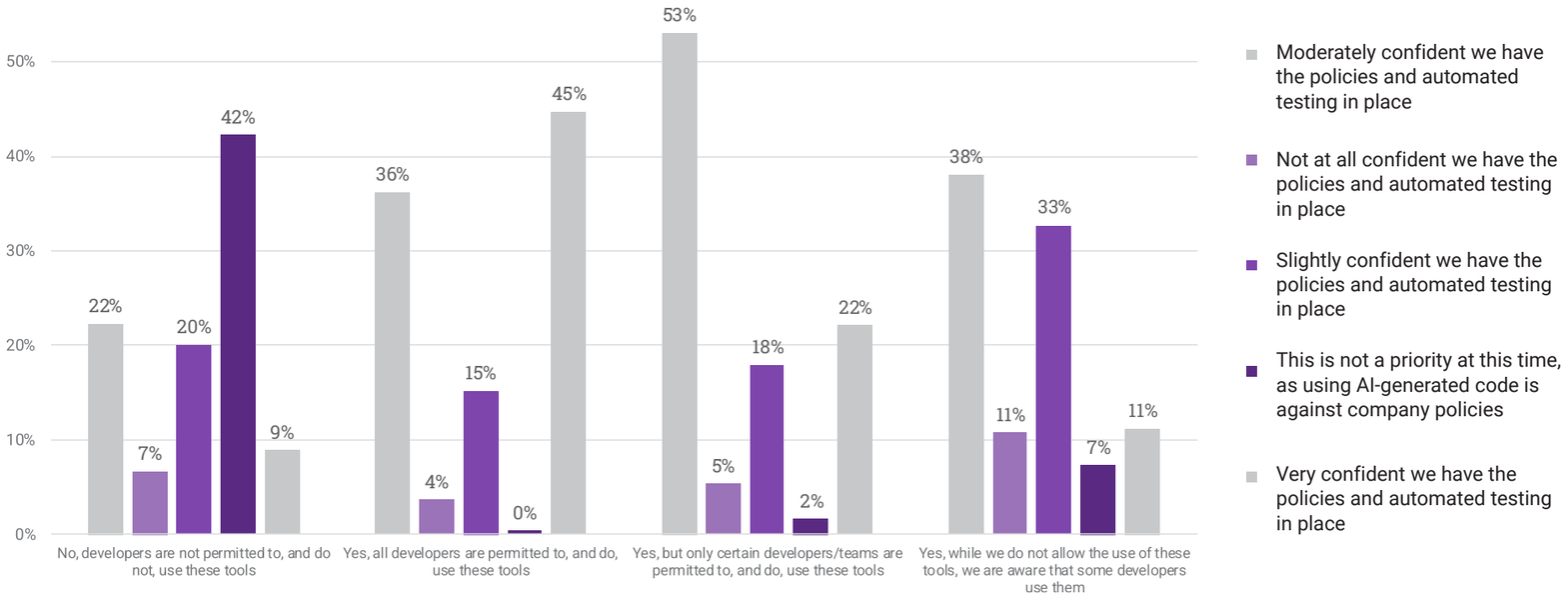
Confidence in security controls amid AI development

In Figure 1, starting from the left, less than 5% of organizations forbid developers from using AI to write code or modify projects. Perhaps this group's moderate and high confidence in their preparedness derives from their prohibition of the use of AI, or perhaps there are other access controls that preclude access to AI resources.

The second group, 27% of respondents, reports a strong awareness that AI is being used. Eighty-one percent have moderate or high confidence in their security preparedness (22% of overall responses). These respondents are readily leveraging AI tools and confident that they have the controls in place to mitigate consequent risks.

The third and fourth groups are in the midst of an AI evolution, with moderate to high confidence in their security preparedness and a seemingly phased approach to AI-enabled development.

Figure 2. Developers' AI usage (permitted or not) correlated against low to slight confidence in security controls



<5% of orgs **27% of orgs** **43% of orgs** **21% of orgs**

In Figure 2, we can see some dissonance between respondents' use of AI-generated code and AI-assisted development, and the steps they're taking to safeguard their intellectual property and mitigate security risks.

Starting from the left, the less than 5% that forbids the use of AI tools altogether exhibits slight or nonexistent confidence in security preparedness, with nearly 42% of this group claiming a lack of priority. Consequently, their choice to disallow AI-enabled development may stem from this lagging organizational approach to securing AI-generated code.

The rightmost group highlights a greater exposure to risk, where automated testing of AI-generated code is a notably lower priority despite an awareness of the use of AI-assisted development.

The group second from right illustrates a seemingly phased adoption of AI-enabled development and security controls, with limited permission being granted, perhaps based upon a slight confidence in preparedness.

Most concerning is the group second from left, which has some development teams that are using AI with permission, despite a clear lack of confidence in their preparations to mitigate risks.

AI and code snippets

A common practice of developers is to use "snippets" (small extracts from larger pieces of code) in software, a problem now exacerbated by the use of AI coding assistants. Although code might include only a snippet of open source, users of the software must still comply with any license associated with the snippet.

Even one noncompliant license in software can result in legal reviews, freezes in merger and acquisition transactions, loss of intellectual property rights, time-consuming remediation efforts, and delays in getting a product to market.

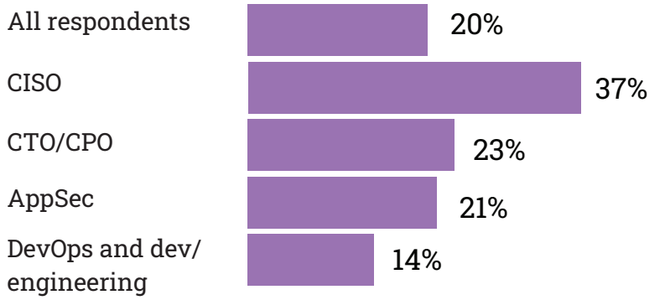
Black Duck's 2024 OSSRA report relates that over half—53%—of the applications examined contained open source with license conflicts, exposing those applications' owners to potential IP ownership questions.

Interpreting and acting on security test results

The effectiveness of application security testing hinges not just on the execution of tests, but also on the ability to interpret results and take appropriate action. This section examines the current state of result interpretation and remediation based on our survey results, highlighting both progress and persistent challenges in the field.

Q7. Which statement best describes the clarity and actionability of the results of your application security tests?

Security test results are extremely easy to understand and to act upon



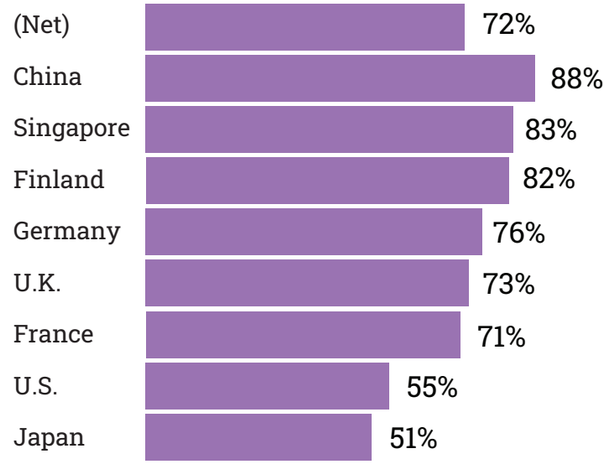
Role-based differences

Our analysis suggests that CISOs, CTOs/CPOs, and AppSec professionals generally reported higher levels of ease in understanding and acting upon security test results compared to other roles (Question 7). For example, 37% of CISOs, 23% of CTO/CPOs, and 21% of AppSec professionals found security test results “extremely easy” to understand and to act upon.

In contrast, only 14% of DevOps and dev/engineering personnel found these tasks extremely easy. This may be due to senior-level personnel having more experience or better interpretative tools at their command than workers in the trenches. Unfortunately, those workers are usually the ones on the front line of security testing and the ones whose efforts are being hampered by the lack of clarity in testing results.

Q7 Which statement best describes the clarity and actionability of the results of your application security tests (by regional)?

Regard results as easy to interpret and act on



Geographical differences

Notable variations were observed across countries. For example, 88% of respondents in China found testing results easy to understand, compared to 55% in the U.S. and 51% in Japan. These regional disparities suggest differences in tool adoption, security culture, or regulatory environments across countries.

Different approaches to parsing and cleansing results

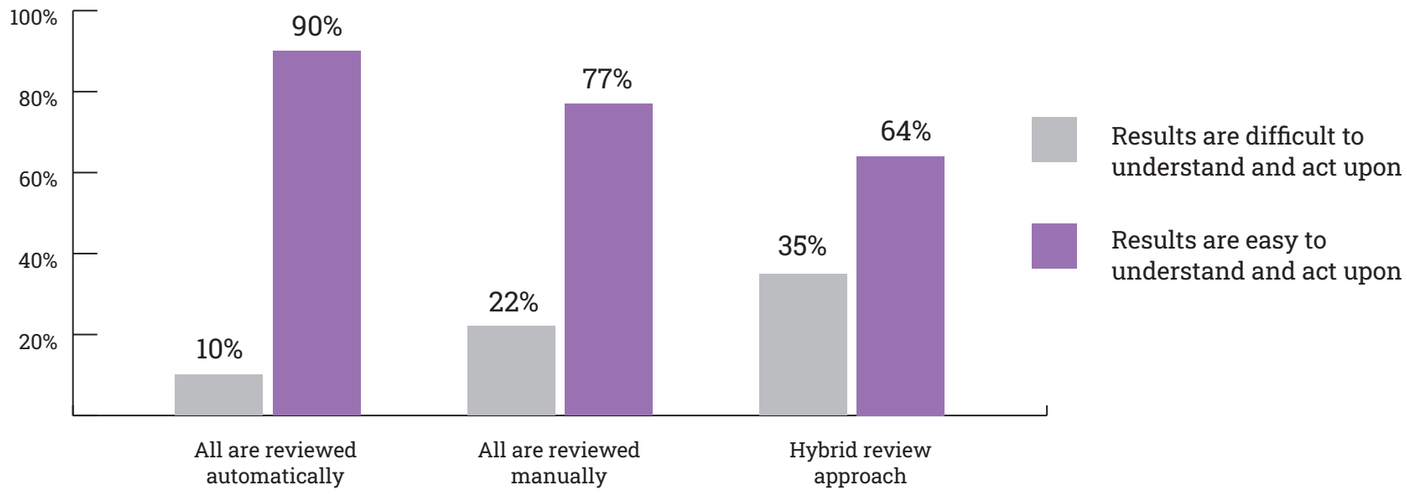
Q8. Which statement best describes your approach to parsing and cleansing the results of application security tests?

Results generated by all tools are manually parsed and cleansed	38%
We can automatically parse and cleanse results from some testing tools; the remainder are manually parsed and cleansed	28%
Results generated by all tools are automatically parsed and cleansed	25%

The process of parsing and cleansing security test results reveals a spectrum of approaches (Question 8). For example, 38% of respondents manually parse and cleanse results from all tools. Twenty-five percent report fully automated parsing and cleansing of results. Twenty-eight percent use a combination of automated and manual parsing and cleansing.

The prevalence of manual and hybrid approaches (66% combined) indicates a significant opportunity for increased automation and normalization in results processing. However, the challenge lies in balancing automation with the need for human expertise in interpreting complex security contexts.

Figure 3. Impact of review method on understanding results and taking action



Automated vs. manual review

As illustrated in Figure 3, it is possible to associate ease of interpretation and action with the method of parsing and cleansing data. The resulting insight reveals a clear benefit to establishing automated mechanisms for parsing and cleansing security test data, whether the benefit comes from accelerated review or more consistent elimination of noise before human consumption. Of those that manually parse and cleanse test results, 22% find those results somewhat or extremely difficult to understand and act upon. Of those that use automated means, only 10% find the same difficulty.

Conversely, 90% of those that use automated methods to parse and cleanse data find the results of security tests somewhat or extremely easy to understand and act upon, while only 77% report the same ease by doing so manually. Notably, when examining those with hybrid approaches to reviewing test results, we see a "worst of both worlds" experience, with 35% citing difficulty understanding and acting on results, and only 64% finding it easy to do so.

From interpretation to action

Constant security testing vs. development speed tension

Q13. Which statement best describes the relationship between application security testing and software development/delivery?

Application security testing severely slows down development/delivery	18%
Application security testing moderately slows down development/delivery	43%
Application security testing slightly slows down development/delivery	25%
Total	86%

Despite advancements in tools and processes, tension remains between thorough security testing and the need for development speed, as shown in the responses to Question 13. Eighty-six percent of respondents feel that security testing slows down development by some amount (ranging from slightly to severely). The plurality (43%) feels that testing moderately slows down development. While one-quarter of respondents feel that security testing slightly slows down development/delivery, and another 18% feel that it severely slows the development life cycle.

There may be more insight, though, in looking at how software projects are added to the security testing queue and whether that is an impediment to development and delivery pipelines. Of those that report security testing severely slows down their pipelines, 33% manage their test queues entirely manually, compared to 17% that manage pipelines entirely through automation.

These statistics underscore the ongoing challenge of integrating security seamlessly into fast-paced development cycles without becoming a bottleneck.

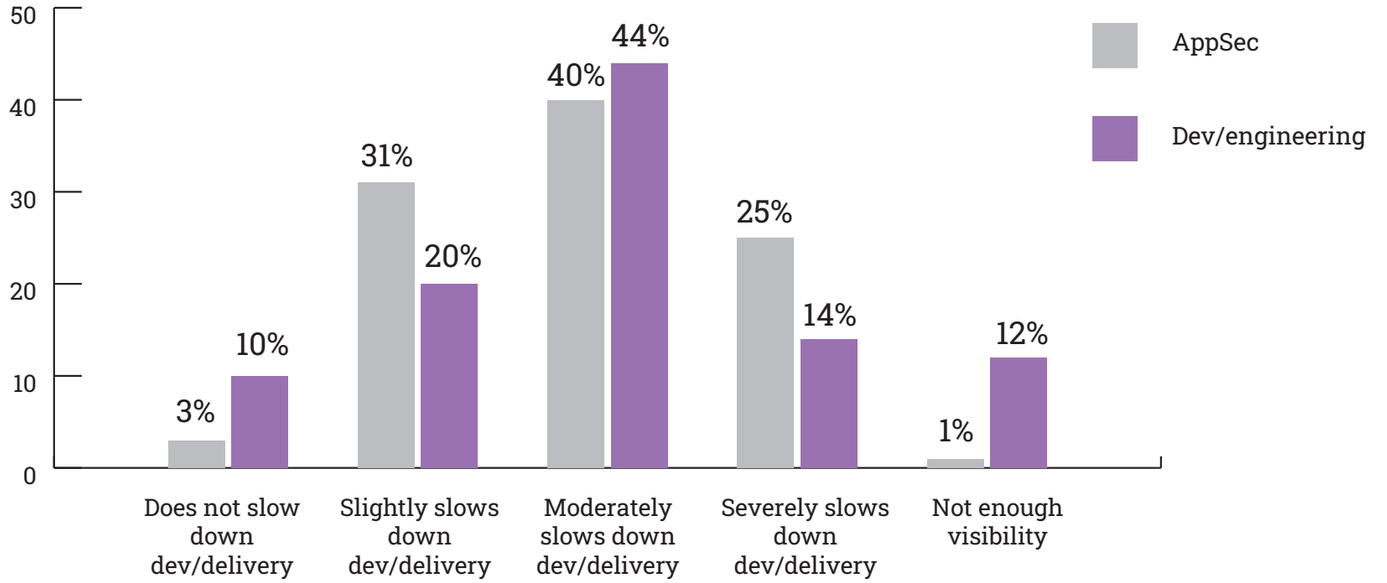
Role-based differences

When examining potential differences in security testing's impact on development and delivery pipelines, there are a few clear distinctions among roles, depicted in Figure 4.

AppSec teams, perhaps due to their proximity to the testing process or the pressures applied to them to accelerate review, show the greatest sentiment that tests moderately or severely impede pipelines (65%).

Similarly, 58% of dev/engineering personnel share this sentiment. It's important to note that visibility into security testing is a significant challenge for dev/engineering teams, making it likely more difficult for them to assess the impact of security tools. This can make a concerted DevSecOps initiative more difficult to implement, as critical contributors are unable to close feedback loops and optimize efforts appropriately.

Figure 4. AppSec and dev/engineering perception of security testing's impact on development/delivery



Organizations are actively implementing automated security measures throughout the development life cycle, with a focus on communication, prevention, and integration with existing workflows. However, there's still significant room for wider adoption of these practices.

Let's now extend each role's perception of pipeline impediment to include the method of managing the security testing queue. We can validate that each role benefits from automating security testing. When manually managing testing queues, 29% of dev/engineering personnel and 44% of security personnel feel severe impact on development and delivery timelines. When managing testing queues through automation, only 16% of dev/engineering personnel and 19% of security personnel feel a severe impact to development speed.

This illustrates a great benefit to development and delivery pipelines, yet also defines a consistent perception among dev/engineering teams that security testing tends to negatively impact their workflows. Ultimately, dev/engineering teams report only a 13% reduction in perceived slowdown, whereas security teams report a 25% reduction.

How remediation is accomplished

A major goal of security testing is to drive remediation efforts. Our survey reveals several key aspects of the remediation process among respondents.

The responses to Questions 10 and 11 indicate that organizations are actively implementing automated security measures throughout the development life cycle, with a focus on communication, prevention, and integration with existing workflows. However, there's still significant room for wider adoption of these practices.

Prioritizing issues for remediation

Q10. Which statement best describes your approach to prioritizing detected security issues for remediation?

Issues are automatically prioritized for remediation based on policies/risk tolerance	49%
Issues are manually prioritized for remediation	43%

Question 10 shows that nearly half the surveyed organizations are using automated systems to prioritize security issues, indicating a significant adoption of advanced risk management practices. But a substantial portion (43%) still rely on manual prioritizations. The close split between automated and manual prioritization suggests that the arena of software security testing is in a transition phase, with many organizations likely using a hybrid approach.

What happens when security issues are discovered

Q11. What actions/mechanisms occur automatically as a result of application security testing results or policy violations?

Alerting to upstream contributors (e.g., developers, engineers, architects)	38%
Assignment to developers via issue management workflows (e.g., Jira, Slack)	36%
Alerting to downstream stakeholders (e.g., security team, partners, customers)	32%
Prevent checking-in of code to SCM/repositories	32%
Prioritization for triage and remediation	32%
Prevent addition of compiled assets into binary repositories	30%
Block promotion into staging/production	28%
Breaking the build	24%

The responses to Question 11 reveal that organizations are employing a variety of automated actions to address security issues, indicating a mature, layered approach to security.

The top actions involve alerting various stakeholders (38% for upstream contributors, 32% for downstream stakeholders), emphasizing the

The top five methods of assigning remediation issues are all automated, indicating a strong trend toward automating the notification process. This aligns with broader DevSecOps principles of integrating security seamlessly into development workflows.

importance of communication in addressing security issues at a pace required by DevOps and CI/CD methodologies. High percentages for assignment via issue management tools (36%) reveal a focus on the DevSecOps requirement for closed feedback loops between security and development teams to accelerate remediation. Significant percentages for actions such as preventing code check-ins (32%), blocking promotion downstream (28%), and breaking builds (24%) demonstrate a shift toward using automated, preventive security measures to preclude risks and avoid realizing exploitable conditions in production environments.

However, while adoption of these automated actions is significant, there's still room for growth, as no single action is implemented by more than 38% of organizations.

How developers are informed of issues

Q12. Out of the following, how are developers/software engineers in your organization notified of/assigned application security issues for remediation?

Automated message via communication/collaboration tools (e.g., email, Microsoft Teams, Slack)	42%
Automated alerts within the security tool (e.g., in-app notification, dashboard)	40%
Automated alerts/assignment within issue management tools (e.g., Jira, Trello)	39%
Automated alerts/logs within development tools (e.g., IDE)	36%
Automated alerts/logs within pipeline tools (e.g., build, SCM, repos)	35%
Manual assignment (e.g., by manager or team lead)	32%

In the responses to Question 12, the top five methods of assigning remediation issues are all automated, indicating a strong trend toward automating the notification process. This aligns with broader DevSecOps principles of integrating security seamlessly into development workflows.

The prevalence of alerts within development tools (36%) and pipeline tools (35%) indicates an attempt to help developers fix issues more quickly. There is a high percentage of responses citing alerts within issue management tools (39%) and security tools (40%), which indicates multiple locations to access necessary risk information. This creates unnecessary deviations from development workflows. While not as common as automated methods, manual assignment of issues is still used by a significant portion (32%) of organizations.

Conclusion

As we conclude this examination of the current state and future trajectory of application security, it's clear that DevSecOps is at a critical juncture. Our findings reveal both progress and problems in current DevSecOps practices.

Over 60% of respondents report that security testing moderately or severely slows down development, highlighting the ongoing challenge of integrating robust security practices without impeding agility. Over 80% of organizations use between 6 and 20 security testing tools, indicating a complex testing environment that can lead to integration challenges, noise, and alert fatigue. In fact, 60% of respondents report that anywhere from 21% to 60% of their security test results are noise, underscoring the need for more-effective filtering and prioritization mechanisms.

While 49% of organizations now use automated prioritization for security issues, reflecting a growing trend toward leveraging technology to streamline security processes, a significant number of respondents are still using manual processes in various aspects of their application security testing and remediation workflows.

With over 90% of organizations using AI tools in some capacity for software development, we're witnessing a transformative shift in how applications are built and secured. This adoption brings both new capabilities and new security considerations. While adoption is high, only 24% of respondents are very confident in their policies, management, and testing for AI-generated code, indicating an area in dire need of automated processes.

The truth of the matter is that, while AI-assisted coding may be accelerating development, security processes—which are already struggling to keep up—are going to fall further behind without automation.

Take the time now to critically evaluate your organization's approach to software security testing.

Over 60%

of respondents report that security testing moderately or severely slows down development

49%

organizations now use automated prioritization for security issues

Over 90%

of organizations are using AI tools in some capacity for software development

Scrutinize your tool stack. Are you drowning in a sea of disparate solutions, or leveraging an integrated, streamlined suite of security tools?

- Evaluate your tools and processes. Aim for consolidation and results integration.
- Reduce tool proliferation and complexity where possible by choosing a primary vendor with the experience and knowledge to consolidate disparate tools into a comprehensive testing whole.
- Explore implementing an application security posture management (ASPM) solution to integrate tools, automate workflows, and normalize and prioritize results. Invest in tools and processes that consolidate security test results and make them more actionable and easier to understand across all roles in the organization.

Evaluate your automation levels. Our survey indicates that manual processes still dominate in many organizations. Identify where automation can be leveraged to boost speed, efficiency, and consistency.

- Explore integrating automated security checks into your CI/CD pipeline.
- Consider implementing infrastructure-as-code (IaC) with built-in security policies.
- Provide developers with IDE plugins for real-time security feedback and prioritized remediation guidance to help them fix faster and cultivate their security capabilities.

Establish AI governance today. Establish clear policies and procedures for the use of AI in software development. Invest in tools and processes designed to vet and secure AI-generated code.

- Static application security testing (SAST) is highly effective at identifying coding flaws early in the development process. This is crucial for vetting AI-generated code, which can inherit insecure coding flaws from its training data.
- Similarly, examining AI-created code with a software composition analysis (SCA) tool can help developers identify and secure outdated or insecure third-party components, as well as open source libraries with licenses that may potentially conflict with an organization's business goals for its software.
- Dynamic application security testing (DAST) detects vulnerabilities at runtime and verifies issues' exploitability. DAST scans are a critical component of application security testing, and the rise in AI-generated code only further highlights its importance. AI coding tools are trained using publicly accessible code repositories, for better or for worse. They are great for generating code quickly, but do not apply the same contextual reasoning a developer would to determine the best way to write code for a specific application. While teams can provide this context to AI tools in the form of prompt engineering, there are still limitations. Ensuring that the application is performing its desired function, and doing so securely, relies on application security testing. Doing so at the speed of AI-enabled development requires that it is tightly integrated into pipelines to allow for the detection of vulnerabilities in the context of the application in its running state.
- Ideally, all three security testing tools—SAST, SCA, and DAST—will run atop a centralized platform or be managed through an ASPM solution. You may also yield greater efficiency and scalability with proper integration and coordination with other AppSec testing tools, developer tools, and issue-trackers you are using.
- Protect sensitive data used to train AI models by ensuring that only authorized personnel have access. Encrypt data both at rest and in transit to prevent unauthorized access. Enforce the principle of least privilege to grant AI systems the minimum access necessary to perform their functions.

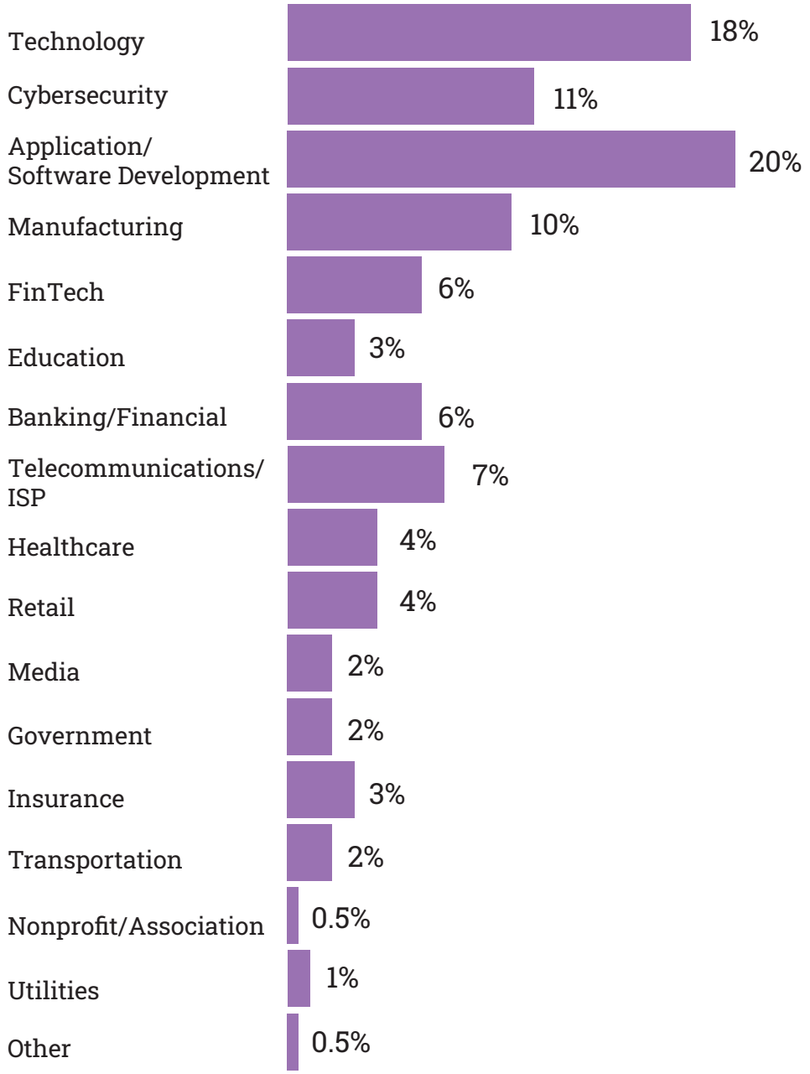
Successful organizations will be those that view the challenges outlined in this report not as obstacles, but as opportunities for transformation and improvement. They'll be the ones not just willing to adapt to the changing landscape of DevSecOps but determined to shape it. The future of DevSecOps is not predetermined—it's waiting to be defined.

What role will you play in the future of DevSecOps?

Appendix

Survey Respondents

Industries of Survey Respondents



Job Roles of Survey Respondents

CISO	16%
CTO/CPO	17%
InfoSec	14%
AppSec	11%
Dev/engineering	22%
DevOps	13%
Cloud ops	4%
QA/testing	2%
None of the above	1%

Survey Respondent Country

U.K.	12%
U.S.	13%
France	13%
Germany	13%
Finland	12%
China	12%
Singapore	12%
Japan	13%

Organization Headcount

Fewer than 100	4%
100–500	11%
501–1,000	13%
1,001–2,000	16%
2,001–5,000	17%
5,001–10,000	18%
10,001–15,000	10%
15,001–50,000	7%
50,001–100,000	14%

Questions

Q1. Which of the following criteria does your organization consider when determining which application security tests to run and when they are run?

Sensitivity of information accessed/transmitted by the application	36.77%
General best practices recommended by third-party organizations (e.g., OWASP)	35.88%
Ease-of-configuration or automation of the security tests	35.38%
Industry requirements or regulatory compliance	34.99%
The application's production environment	33.99%
Attestation of security processes to stakeholders (e.g., customers, partners, investors)	33.70%
Business criticality of the application	32.80%
Release frequency or shipping deadline of the application	30.82%
Recent publication of new vulnerabilities or zero-days	29.34%
None of the above	2.78%

Q2. Which statement best describes your process of configuring and running application security tests across your SDLC or CI pipeline?

Testing tools provided by the same vendor are configured using a centralized interface and automatically run with policies	29.53%
All tests are configured using a centralized interface and automatically run with policies	25.77%
Each test is configured using its own interface and automatically run with policies	22.20%
Each test is configured using its own interface and manually run	15.46%
I am not involved with configuring or running application security tests	4.16%
None of the above	2.87%

Q3. Which of the following statements best describes the manner in which new projects, branches, or repositories are added to your application security testing queue?

All are added to the test queue manually (e.g., declared by dev team, selected by security team)	28.74%
All are added to the test queue automatically (e.g., detected by testing tools)	38.35%
Most are added to the test queue automatically; a few are added manually	22.40%
Most are added to the test queue manually; a few are added automatically	6.14%
I am not familiar with how items are added to the security testing queue	4.36%

Q4. Approximately what percentage of your projects, branches, and repositories are being included in your application security testing queue?

Percentage of projects, branches, and repositories included in testing queue	Percentage of respondents
Up to 20%	4.86%
21%–40%	23.39%
41%–60%	36.77%
61%–80%	20.52%
81%–100%	8.72%
I do not have enough visibility to approximate test coverage	5.75%

Q5. Which of the following teams/departments determine which application security tests are performed, when, and on which projects?

Security	44.00%
Development/software engineering	42.22%
DevOps	36.57%
Quality assurance	33.99%
Compliance	28.15%
Cross-functional groups	21.01%
Legal	19.43%
None of the above	1.39%

Q6. Approximately how many application security testing tools does your organization use? This should include all means to detect software vulnerabilities, noncompliance with security standards, sensitive data leakage, and policy violations.

Number of security testing tools	Percentage of respondents
1–5	9.32%
6–10	33.50%
11–15	33.30%
16–20	14.57%
21+	3.87%
I do not have enough visibility to estimate the number of testing tools	5.45%

Q7. Which statement best describes the clarity and actionability of the results of your application security tests?

Easy (Net)	72.25%
Security test results are extremely easy to understand and to act upon	20.22%
Security test results are somewhat easy to understand and to act upon	52.03%
Security test results are somewhat difficult to understand and to act upon	17.54%
Security test results are extremely difficult to understand and to act upon	4.26%
I am not involved with interpretation or action upon the results of application security tests	5.95%
Difficult (Net)	21.80%

Q8. Which statement best describes your approach to parsing and cleansing the results of application security tests?

Results generated by all tools are manually parsed and cleansed	38.06%
We can automatically parse and cleanse results from some testing tools; the remainder are manually parsed and cleansed	28.05%
Results generated by all tools are automatically parsed and cleansed	25.27%
I am not involved with parsing and cleansing application security test results	5.35%
None of the above	3.27%

Q9. Approximately what percentage of security test results are noise? For example: duplicative results, false positives, conflicting with other tests/tools.

Percentage of noise in findings	Percentage of respondents
0%–20%	15.06%
21%–40%	30.23%
41%–60%	30.23%
61%–80%	14.87%
81%–100%	2.78%
I do not have enough visibility into all tests and results to identify noise	6.84%

Q10. Which statement best describes your approach to prioritizing detected security issues for remediation?

Issues are automatically prioritized for remediation based on policies/risk tolerance	49.45%
Issues are manually prioritized for remediation	42.62%
I am not familiar with the process of prioritizing issues	7.93%

Q11. What actions/mechanisms occur automatically as a result of application security testing results or policy violations?

Alerting to upstream contributors (e.g., developers, engineers, architects)	37.66%
Assignment to developers via issue management workflows (e.g., Jira, Slack)	35.58%
Alerting to downstream stakeholders (e.g., security team, partners, customers)	32.11%
Prevent checking-in of code to SCM/repositories	32.41%
Prioritization for triage and remediation	31.81%
Prevent addition of compiled assets into binary repositories	29.93%
Block promotion into staging/production	28.44%
Breaking the build	23.89%
No actions or mechanisms are automated, all are manual based on test results	4.96%
Other	0.20%

Q12. Out of the following, how are developers/software engineers in your organization notified of/assigned application security issues for remediation?

Automated message via communication/ collaboration tools (e.g., email, Microsoft Teams, Slack)	41.82%
Automated alerts within the security tool (e.g., in-app notification, dashboard)	40.14%
Automated alerts/assignment within issue management tools (e.g., Jira, Trello)	39.35%
Automated alerts/logs within development tools (e.g., IDE)	35.88%
Automated alerts/logs within pipeline tools (e.g., build, SCM, repos)	34.69%
Manual assignment (e.g., by manager or team lead)	32.11%
I am not familiar with how developers/ engineers are made aware of security issues	3.27%
None of the above	1.98%

Q13. Which statement best describes the relationship between application security testing and software development/ delivery?

Application security testing severely slows down development/delivery	18.04%
Application security testing moderately slows down development/delivery	42.81%
Application security testing slightly slows down development/delivery	24.68%
Application security testing does not slow down development/delivery	9.22%
I do not have enough visibility to assess the relationship accurately	5.25%

Q14. Are your developers using AI, generative, or transformational tools to write code and modify projects?

Yes (Net)	90.29%
Yes, all developers are permitted to, and do, use these tools	26.86%
Yes, but only certain developers/teams are permitted to, and do, use these tools	42.91%
Yes, while we do not allow the use of these tools, we are aware that some developers use them	20.52%
No, developers are not permitted to, and do not, use these tools	4.66%
I do not have enough visibility into development processes to know if these tools are used	5.05%

Q15. How confident are you that you have the processes in place to manage and secure AI-generated code?

Confident (Net)	84.94%
Very confident we have the policies and automated testing in place	24.08%
Moderately confident we have the policies and automated testing in place	41.33%
Slightly confident we have the policies and automated testing in place	19.52%
Not at all confident we have the policies and automated testing in place	6.05%
This is not a priority at this time, as using AI-generated code is against company policies	4.46%
I do not have enough visibility into our processes to manage and secure AI-generated code	4.56%

About Black Duck

Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.