ShiftLeft

ShiftLeft

Developer Productivity & Security Survey

June 2020

Top Insights

- Nearly 70% of software development organizations are releasing multiple times per month or more and 17.7% of organizations daily or faster
- The top 4 ways of integrating security into the SDLC are also the 4 least productive
- 96% Developers believe disconnected security & development workflows inhibit their productivity
- 93% of AppSec professionals believe that poor quality of security scanning/testing results inhibit developer productivity

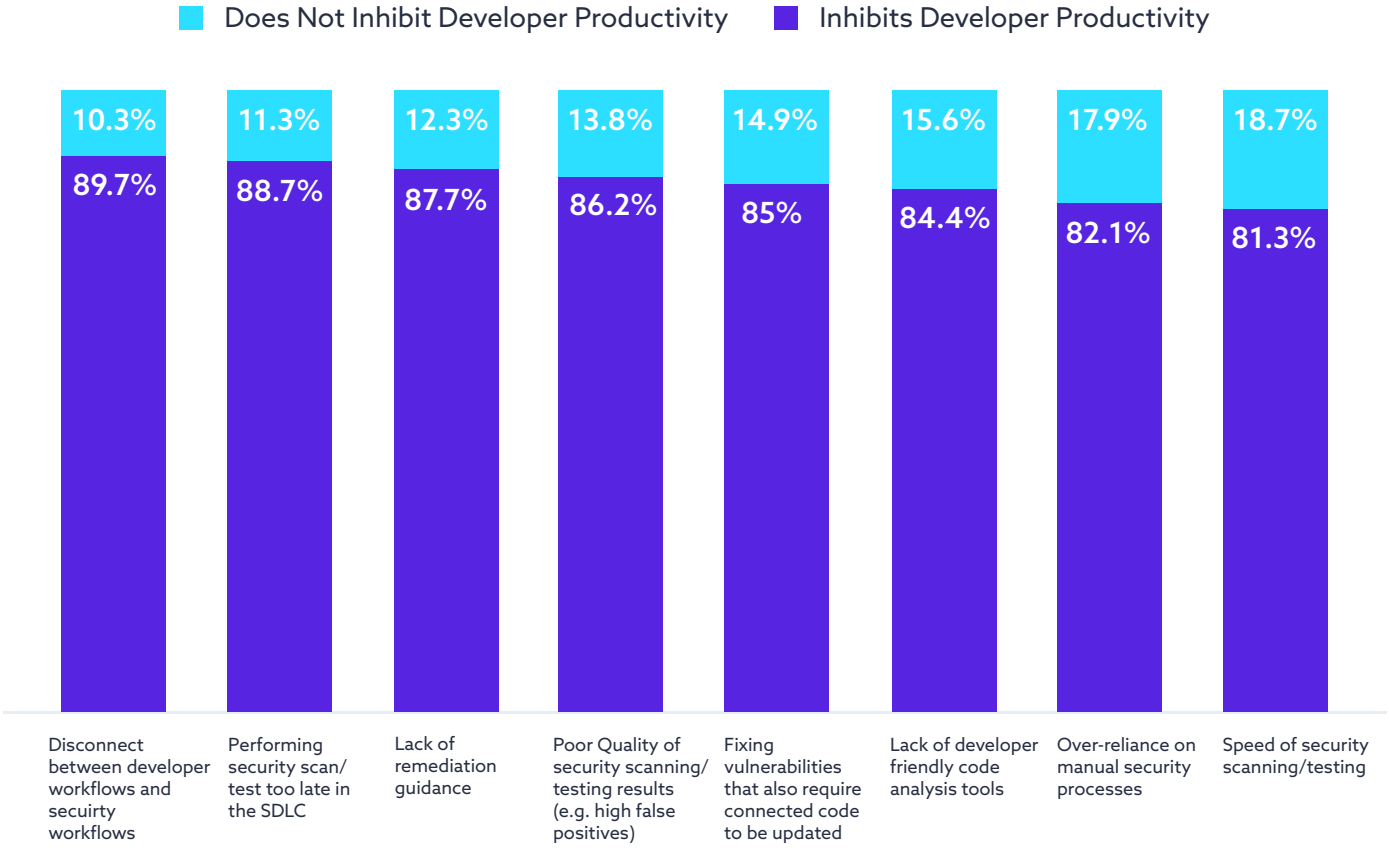
ShiftLeft set out to understand the link between application security (AppSec) and developer productivity. As the software development life cycle (SDLC) modernizes, numerous advances have increased features velocity such as cloud computing, microservices architectures, DevOps, virtualization, containers, agile development methodologies, serverless and orchestration. However, these exponential changes in development have been met by only incremental changes in AppSec. The survey highlights areas in which AppSec needs to improve in order to maximize developer efficiency through the voices of development and security professionals themselves.

It is clear from the survey that many aspects of security are adversely impacting developer productivity. Every aspect of security polled had at least 81% of respondents agreeing that it inhibited developer productivity. When isolating for developer responses only, over 86% agreed that every aspect of application security inhibited productivity. The biggest inhibitor of productivity, as reported by 89% of overall respondents, and 96% of developers, is the disconnect between development and security workflows. The rest of the survey provides insights as to why. Legacy AppSec tools were designed in a different era, for a different purpose and a different user. As the SDLC has sped up, AppSec has not kept pace, and insertion strategies and technology limitations are preventing the deployment of developer-friendly security workflows.

Demographic Summary

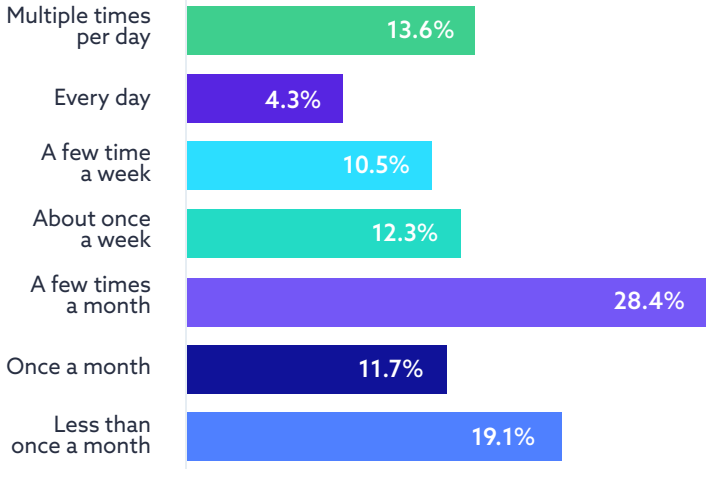
- 165+ Responses, collected April - June of 2020
- The responses were made up of 42% Developers, 30% AppSec, 20% DevOps & 8% other
- The respondents are from 55% North America, 18% Asia & 16% Europe
- The organizations represented are 28% SMB, 40% Mid-Market & 32% Large Enterprise
- The most common programming languages are 65% Java, 56% JavaScript, 53% Python & 35% C#

What are the greatest inhibitors to developer productivity when it comes to application security?



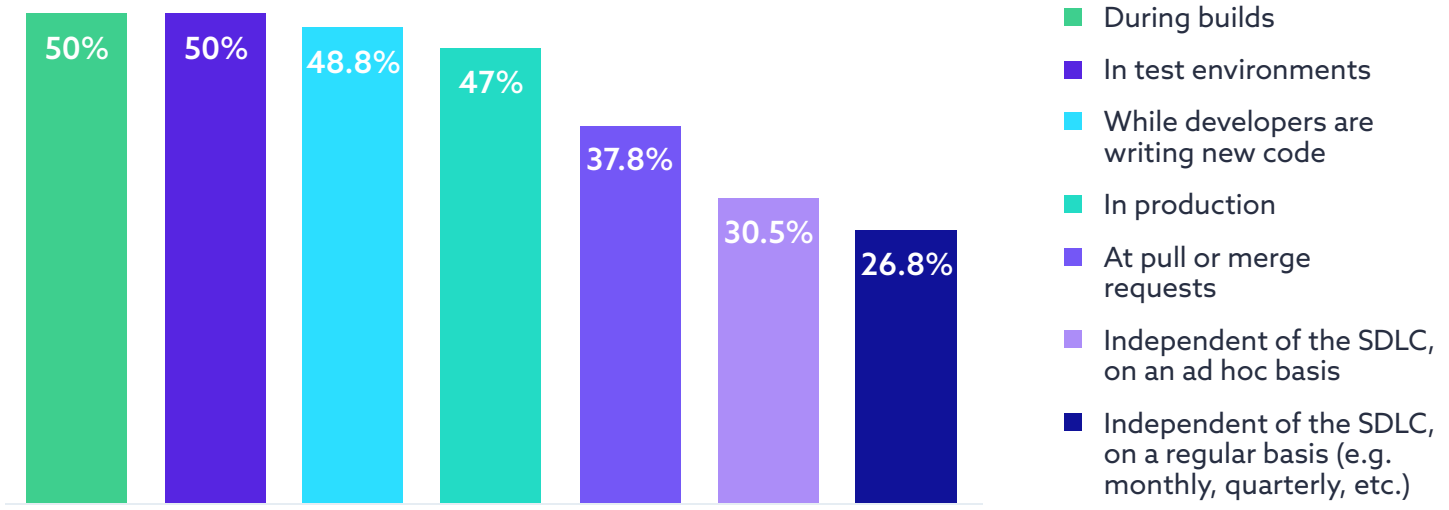
The average software release cycle is now a fast cycle. While the frequency remains a spectrum, fast software release cycles are now more commonplace than slow and regimented waterfall processes. Nearly 70% of software development organizations are releasing multiple times per month or more and 17.7% of organizations daily or faster.

How often does the app(s) that you are responsible for get released?



Compared to the quarterly releases of the waterfall era, the top 13.6% of organizations have increased release velocity by 180X or more. Yet, our survey data demonstrates that AppSec's attempts at catching up have largely focused on stretching pre-existing tools and workflows to meet the new challenge. Inserting security during build, test, production, or in the developer's IDE remains the most common practice.

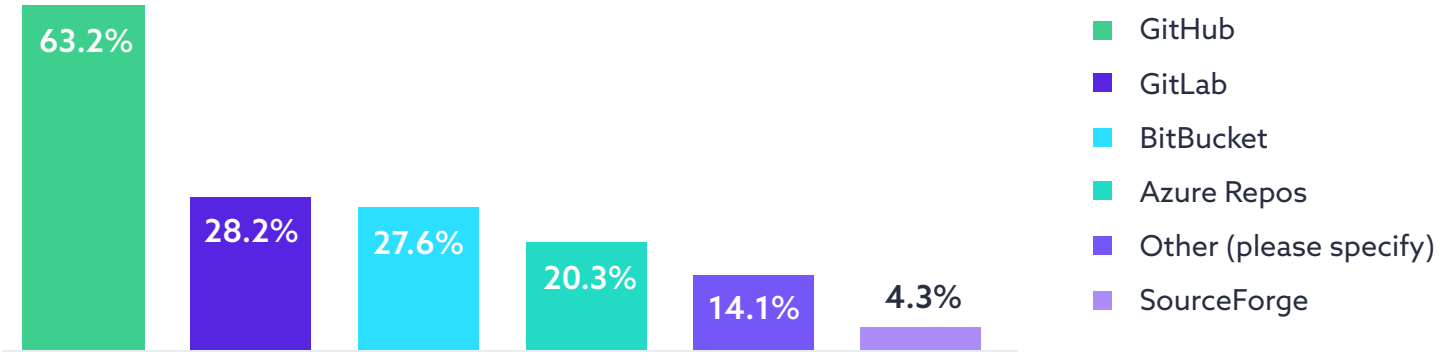
When does your organization secure the app(s) that you are responsible for? (Check all that apply)



However, developers are widely known to dislike (and often disable) IDE-based security tools and the survey also shows that inserting security while developers are writing code to be the biggest inhibitor of developer productivity. Furthermore, build, test, and production may be convenient stages to automate security, but they are inherently disconnected from developer workflows.

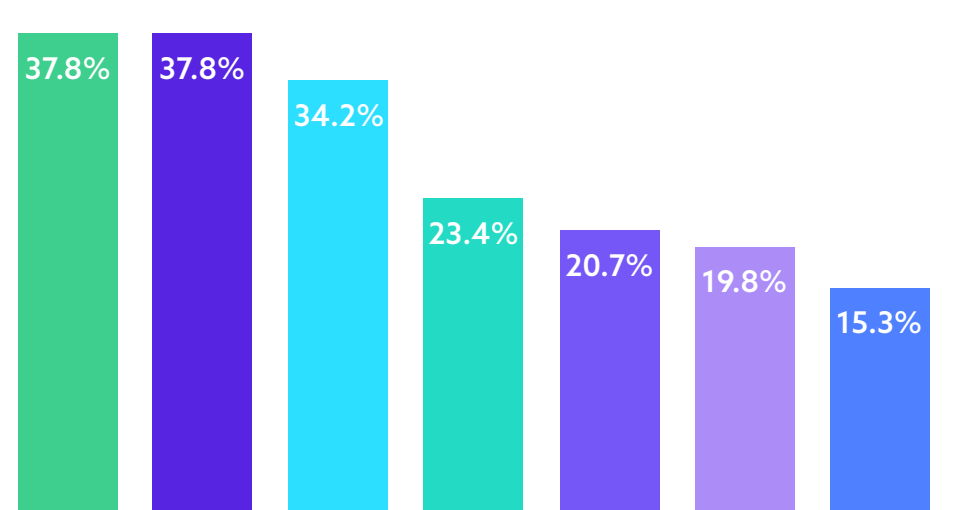
The four most popular code repositories are all distributed version control systems, which proves that modern developer workflows are predominantly git-based.

Which code repository does the app(s) that you are responsible for use? (Please check all that apply)



Despite developers' focus on writing, committing, and merging code, and despite the survey showing it to be the least productivity inhibiting, securing code at the pull/merge request is the least common way to insert security into the SDLC. Hence, it is not surprising that all but 4% of developers report the disconnect between their workflows and AppSec's as an inhibitor of their productivity.

When does your organization lose the most productivity securing the app(s) that you are responsible for? (Please check all that apply)



Protecting the application in production is security's raison d'etre so it's not surprising to see it as the top overall priority. However, the need to create developer-friendly workflows is not lost on AppSec professionals, who rank creating them as their top security priority, ahead of protecting the application in production.

How would you prioritize the following security challenges for the app(s) that you are responsible for?

Our survey data suggest that security tools' shortcomings may be a reason for the persistent workflow disconnect, despite the widely acknowledged need and top prioritization. In fact, 93% of AppSec professionals report that poor quality of scanning/testing results, such as false positives, hurt developer productivity. Scanning speed and lack of remediation guidance are other features that hurt developer productivity.

With developer to AppSec staffing ratios often in excess of 200-to-1, chronic AppSec talent shortages and burnout, it is clear that scaling to meet the needs of the modern SDLC is not something AppSec can spend or hire its way to. Engaging developers and creating a culture of accountability amongst development teams to secure the code they write in a timely manner is the only way security can match the pace of modern development. Productivity remains the stumbling block, but solving it won't be one-size-fits-all.

Different tools will still require different processes. For example, tools like static application security testing (SAST) and software composition analysis (SCA) should be considered developer-centric because their value is derived less in how the tool is operated (traditionally by AppSec teams) and more in how the results are handled (by development teams). On the other hand, dynamic application security testing, penetration testing, and web application firewalls (WAFs) should be security-centric because they require deep security expertise to configure and operate effectively.

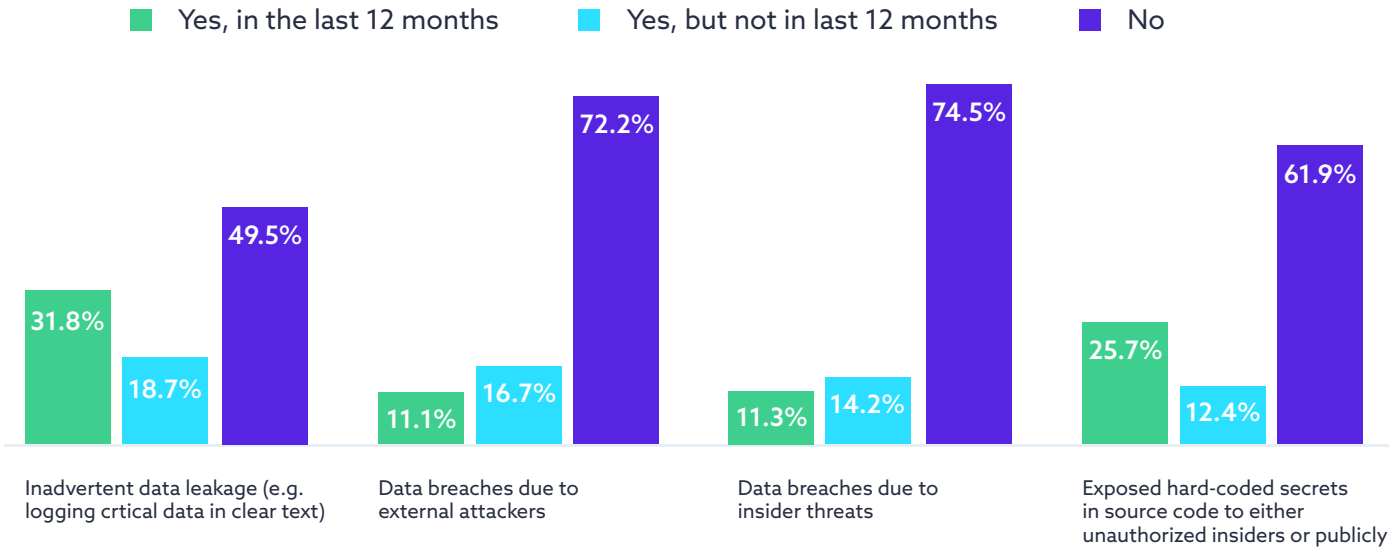
Creating Developer-Centric Workflows with ShiftLeft NextGen Static Analysis

At ShiftLeft, we believe the key to securing the modern SDLC is creating purpose-built developer workflows for developer-centric security tools like SAST and SCA. Our NextGen Static Analysis (NG SAST) combines the industry's fastest code analysis (500,000 lines of code in under 60 seconds) with the most accurate results (highest ever SAST score on the OWASP Benchmark) into a workflow that creates immediate security feedback loops, without forcing the developer to leave their environment. The process is fully automated, synergistic with the developer workflow, and AppSec teams can enforce policy through custom security build rules that approve or deny the pull request.

To try our NG SAST developer-centric workflows in your environment, please create a free, on-demand account by registering with your GitHub credentials (or username and password) at https://www.shiftleft.io/register.

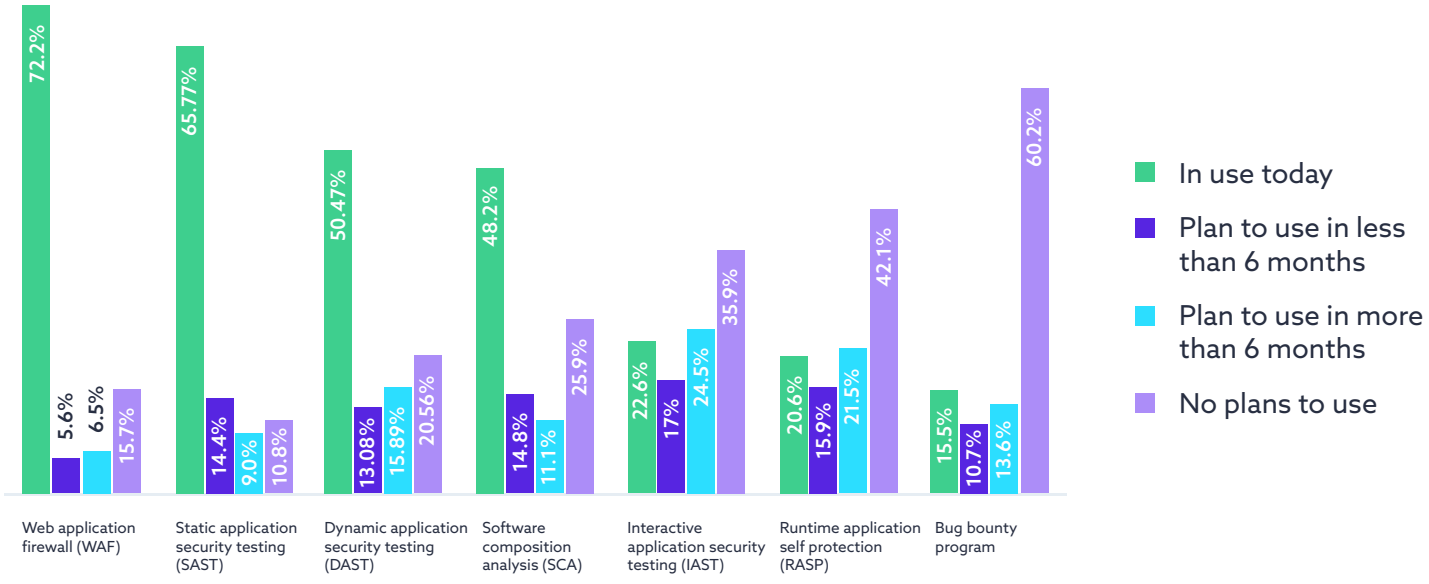
Additional Survey Data & Audience Profile

To the best of your knowledge, have the app(s) you are responsible for had any of the following occur?

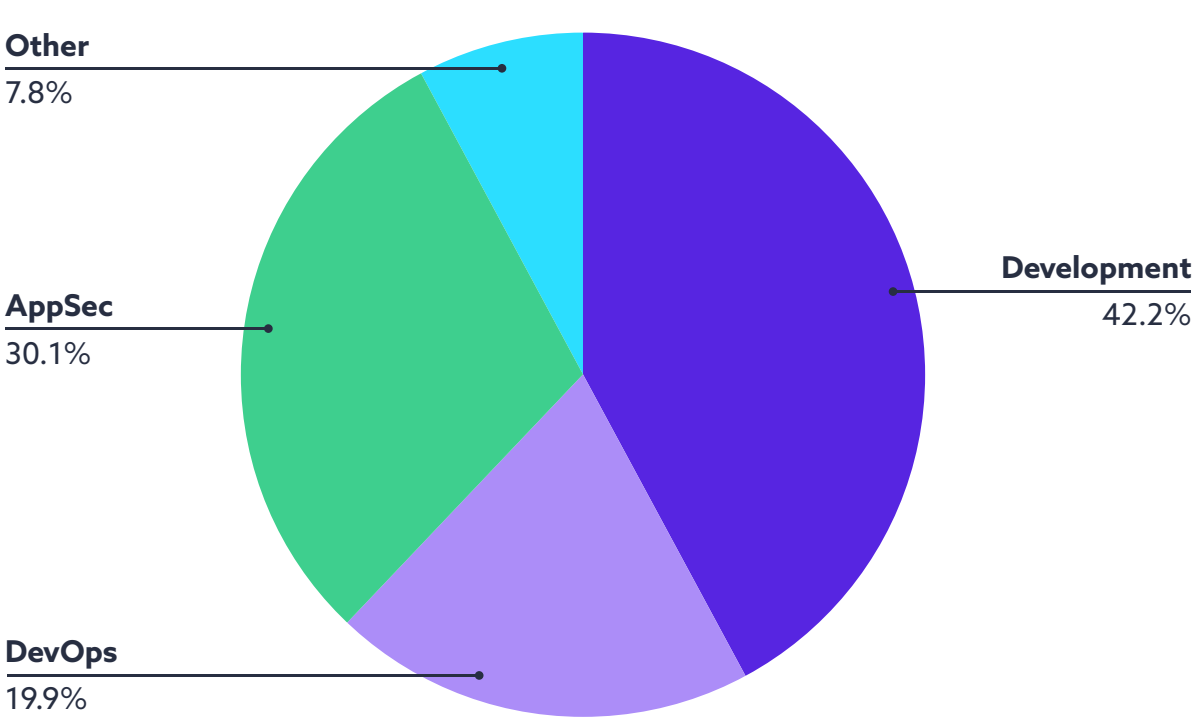


External attackers may get the most attention, but inadvertent data leakage (2.9X) and exposing hardcoded secrets (2.3X) were significantly more likely to have occurred within the last 12 months. This may be because testing for these types of vulnerabilities in development is not yet common and protecting against them in runtime is difficult.

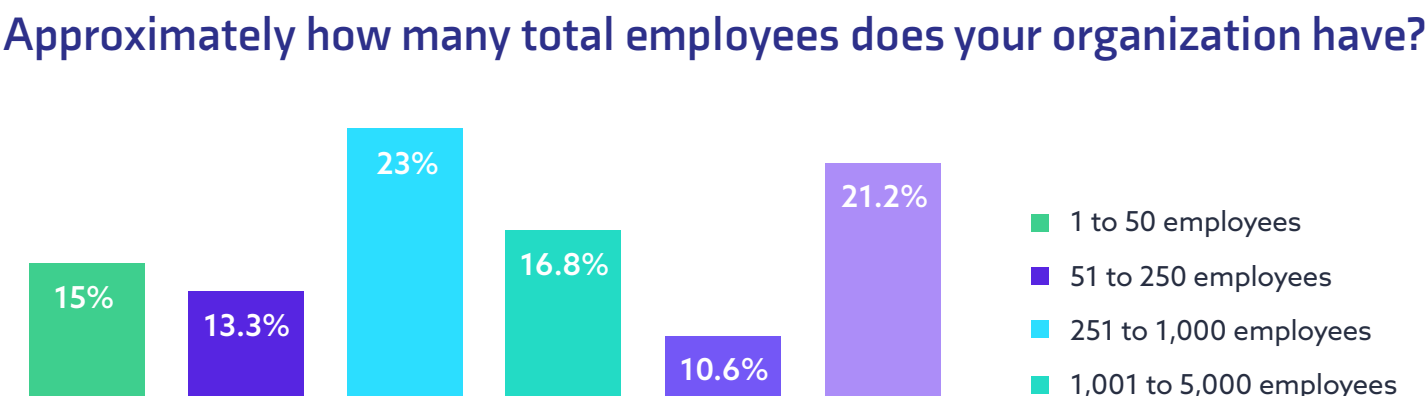
For the app(s) that you are responsible for, what best describes how the following application security technologies are used or not used?



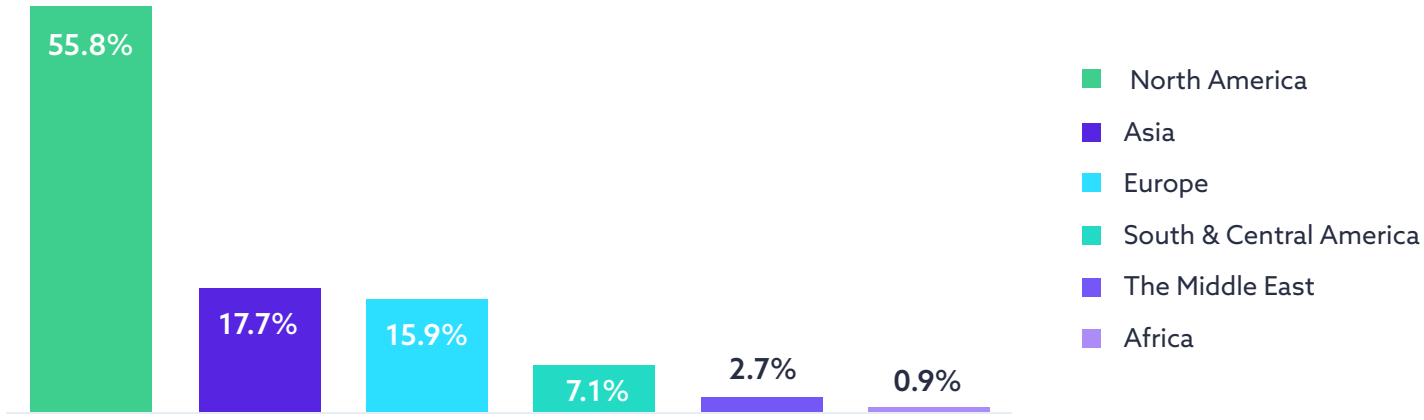
What best describes your primary role regarding the app(s) you are responsible for?



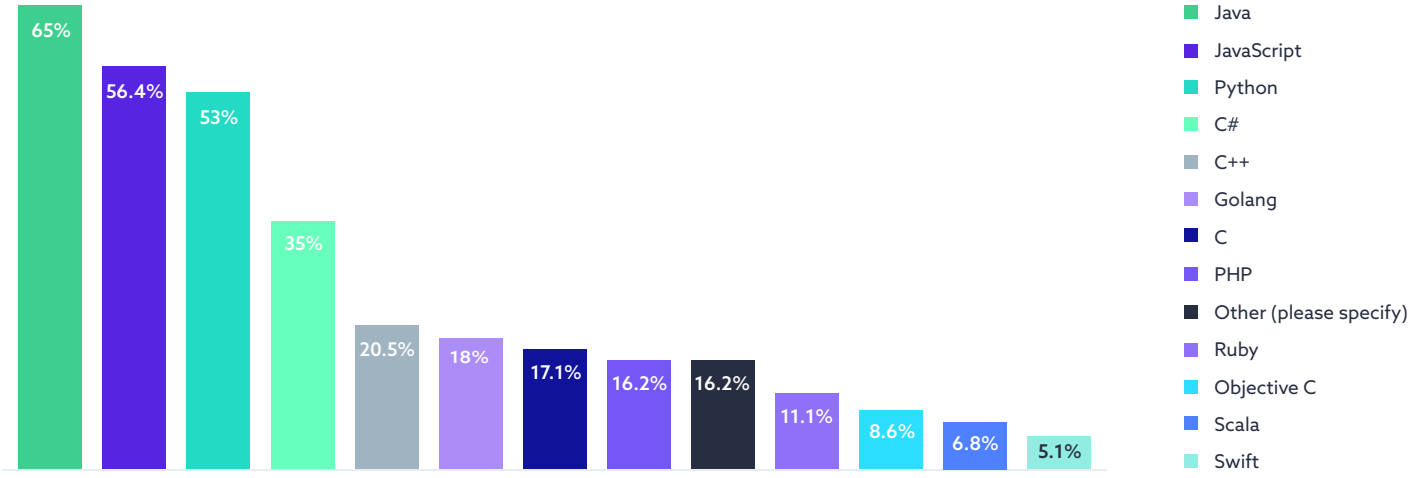
Approximately how many total employees does your organization have?



Where is your organization's headquarters located?



Which languages are the app(s) you are primarily responsible for written in? (Please check all that apply)



While its no surprise to see Java, JavaScript, Python, and C# in the top spots, it is surprising to see Objective-C and Swift in the bottom three, despite the prevalence of mobile applications.

Which build tools does the app(s) you are responsible for leverage? (Please check all that apply)