DevSecOps Insights

2020

powered by snyk

Table of contents

Introduction

Fast software development iterations call for baked-in security in order to keep up with the rate of building and shipping software. In a typical organization, security staff is vastly out-numbered compared to operations and developers. This significantly complicates the job of keeping up with security tests, reviews, etc, in order to mitigate the increasing application security risk.

Is security slowing down operations and developers? This is one of the major concerns and challenges for integrating security in development teams. Security teams remain accountable for the security of applications and related data, yet cannot introduce disruption to the development teams' workflows. To overcome these challenges, development and security teams need to adopt new ways of working together, develop new processes and adopt new tooling. DevOps teams do not prioritize for security in a build pipeline or security monitoring, as there are other concerns they are tasked with. So, even for empowered DevOps teams, security is still mainly an afterthought.

To address security concerns while keeping up with the rapid pace of software delivery, we need to adopt processes, culture, and proper tools through automation which sustains fast development iterations. These enable development teams to integrate security tooling within their build pipelines to detect vulnerabilities early on, and fosters healthy collaboration across security and DevOps teams.

In this report we aim to explore the state of DevSecOps adoption and the challenges organizations and teams face. What we aim to gain from this research is better insight into practices and tools that accelerates DevSecOps adoption.

About the data

This study is based on data presented in the Snyk 2019 State of Open Source Security report and the Puppet 2019 State of DevOps report.

The Snyk report presents the survey results of over five hundred respondents and the Puppet report presents data from 2,949 technical professionals.

Key takeaways - challenges faced in DevSecOps

CULTURE Security is perceived as an activity that slows down the business and overall software delivery. 33% of respondents, within the highest level of security integration, still feel that security is a major constraint on the ability to deliver software quickly.

CULTURE Key security activities, such as threat modelling and security tools integrated in the development pipeline, contribute to a sense of shared responsibility across different functions of the business. Seeing security as a shared responsibility improved by 31% between Level 1 — the lowest level of security integration within an organization — and Level 5, the highest.

CULTURE Even though there is a high correlation between the maturity of security integration and the sense of shared responsibility, 29% of all organizations, positioned at the highest level of security integration, still feel that security teams and delivery teams encounter a lot of friction when collaborating.

CULTURE 81% of users feel developers are responsible for open source security and 68% of users feel that developers should own the security responsibility of their container images.

TOOLING 79% of organizations are positioned at a medium level of DevOps evolution and face challenges in scaling the tooling, culture, and practice to properly support the business.

TOOLING 22 percent of firms at the highest level of security integration are also at an advanced stage of DevOps evolution.

TOOLING 65% of respondents confirm that they employ automated security testing tools to audit their code, while a security code review is an activity that 79% of respondents follow.

TOOLING 37% of users don't implement any sort of security testing during CI.

TOOLING 57% of respondents test for known security vulnerabilities in their open source dependencies, and 36% of respondents perform static application security testing for their own code.

TOOLING 31% of respondents aren't tracking any application dependencies in use within their organization, and 37% are only tracking direct dependencies.

Challenges faced in DevSecOps

As teams and organizations rush to embed security throughout the software development life cycle (SDLC), challenges arise in many forms. In order to cope with the increased pace of software delivery and the scarcity of security resources available, organizations must face cultural, process, and tooling decisions –from shaping internal culture, to finding the right tools to embed in engineering workflows that enable and empower developers and minimize disruptive unplanned work.

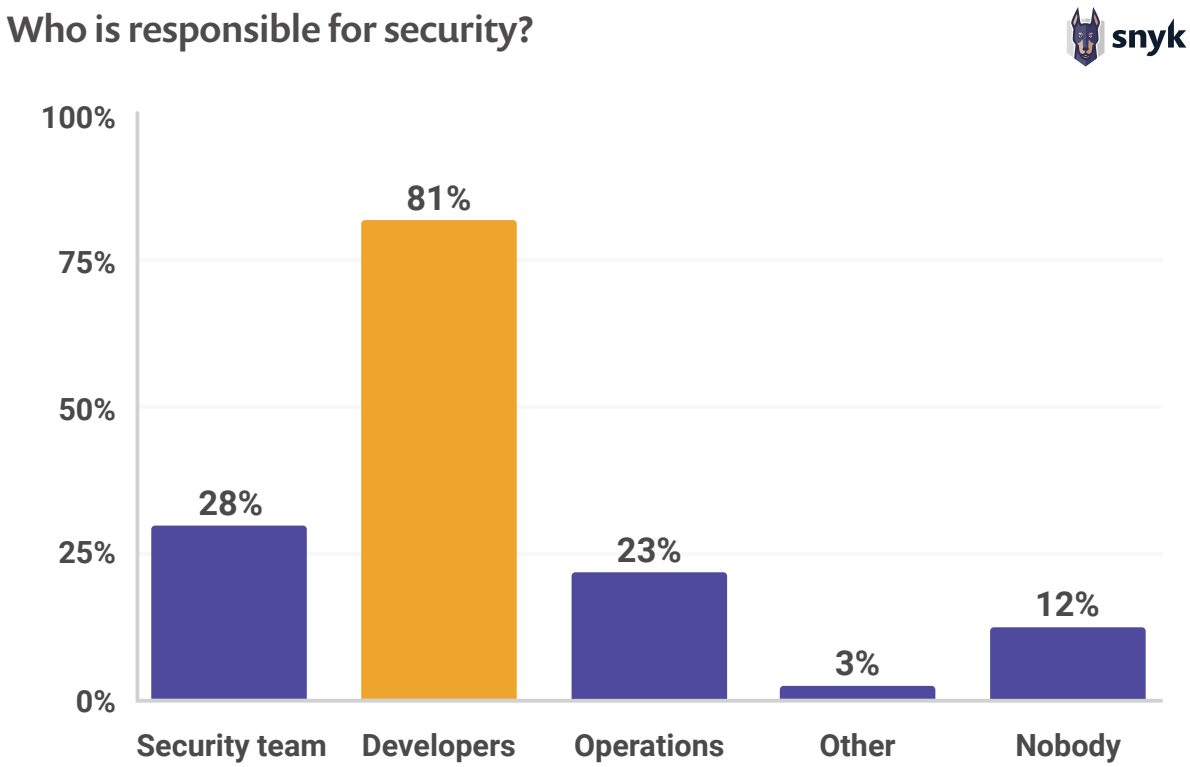
With every data breach disclosed, organizations become more aware of the need to address security early on and throughout the SDLC to ensure customer privacy and assets, feature security, and delivery speed. To do it all well, DevSecOps must be driven by security, but powered by developers.

The culture of DevSecOps

Everyone talks about security as a shared responsibility concept across the organization. Then why is it hard to turn to reality in so many cases? The Puppet State of DevOps report provides a cynical but not uncommon perspective on the matter: security concerns are often viewed as a means of averting a blame rather than measurably improve the overall security posture of an organization.

Moving on to another crucial question, is application security owned solely by the security team? From the Snyk State of Open Source Security report 2019 we learn that 81% of the respondents believe developers should actually own security, but that they aren't well-equipped to do so.

According to this survey, developers are the ones responsible for the security of their code and application.



How can we make them more successful? How can we empower these security champions to better integrate security into their workflows?

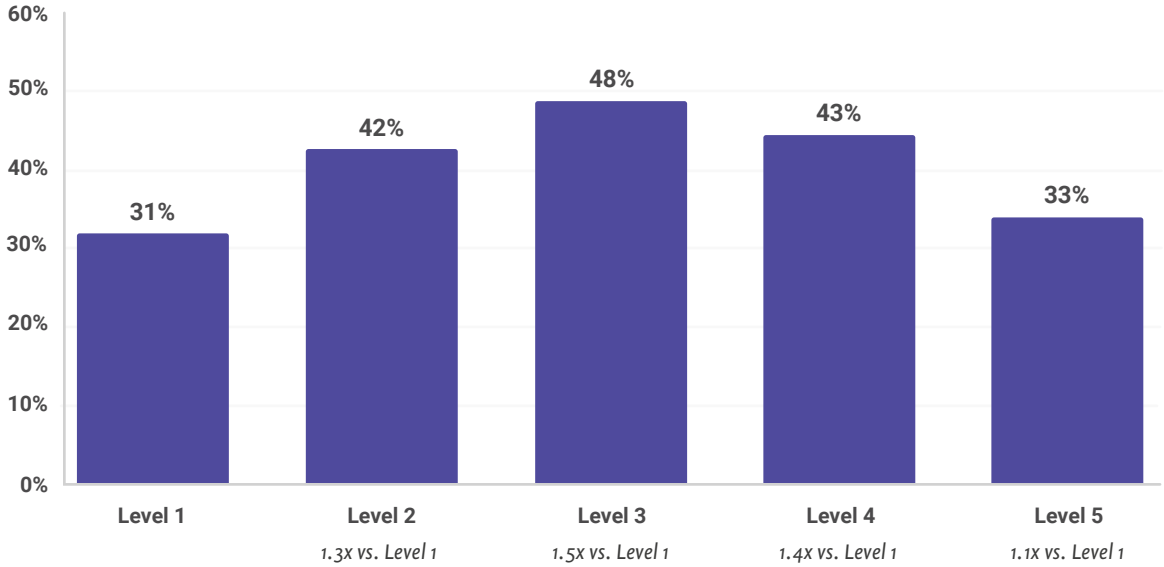
Perhaps, an even more controversial question to ask would be, who is responsible for applying security fixes and who is tasked with finding them?

As DevSecOps processes are being increasingly embraced, some wonder whether security hinders fast development iterations? Unlike DevOps, which is celebrated for speeding up software delivery for agile teams, security is perceived as that which slows it down. Teams need to handle top-to-bottom pressure from business stakeholders who communicate urgency for feature delivery and constantly prioritize it over technical debt and security issues. These issues often remain unaddressed and pose a potential risk of software development slow-down, and paramount business risk.

In fact, according to the Puppet report, 48% of respondents still feel that security is a major constraint on the ability to deliver software quickly.

Security integration and speed of software delivery

Respondents agree security is a major constraint on ability to deliver software quickly



Based on the Puppet "State of DevOps" report

Traditional security practices take place late in the SDLC, for example, moving a release to QA. Even for teams practicing agile software delivery based on short sprints, a security review isn't something that happens often, such as a quarterly review, and isn't baked into the development and automated build and test stages.

Penetration tests and the ability to outsource them, are a handy security tool for many organizations but they take place during the latest part of the SDLC, after the application has been significantly built. At this point, finding severe security issues is costly as a study by IBM System Science Institute suggests. This study shows that fixing a bug is a hundred times more costly when discovered in production. Security bugs further increase the damage beyond just R&D spend, for example those relating to ransomware, loss of data, privacy issues lawsuits, and other risks.

Teamwork and support from the entire organization are critical for successful DevSecOps adoption.

Integrated tooling is key

Software is 'eating the world' and it isn't slowing down, as we're witnessing more and more traditional technologies tunnel into the software world—from software-defined networking to cloud providers that abstract the entire management of traditional services into Infrastructure as Code (IaC).

As software continues to control more of the world around us; developers are key to effectively addressing security concerns because they have a direct impact on code and its security. We're seeing this change being embraced within leading developer ecosystems, such as key security integrations within the largest code repository hosting at GitHub.

To align with the agility demanded in a DevOps world, tooling must incorporate automation from risk detection through remediation. This empowers engineers to proactively address security concerns, leading to rapid risk reduction. Effective tooling should also be contextualized, making it easy for developers to assess and prioritize risk. The volume of *potential risk* at any given time is immense - creating noise for both development and security teams to address. Knowing where the highest risk exists is core to an efficient risk reduction practice for applications and services.

This not only empowers engineers to address security concerns, but also shortens the time window of exposed vulnerabilities, which results in reduced overall risk for the organization and its assets.

IaC flourished within DevOps teams as it enabled them to abstract a lot of the infrastructure management and configuration through a reproducible instruction set. This has the benefit of further tracking via version control, audit logs and such. However, this also implies more access to resources and spinning up infrastructure as needed, which could also suffer from security issues in the form of misconfigured cloud resources, for example, cloud storage, a repeated pattern affecting many cloud data leaks. The case of First American Financial Corp demonstrates this issue with over 885 million records publicly accessible. SImilarly, in the case of the Republican National Committee, vendor data Deep Root Analytics exposed 198 million voter records.

All the above, point to one certain conclusion: security tooling is relevant throughout the entire DevOps stack — from the setup of environments, to their configuration provisioning throughout the application's continuous integration and deployment.

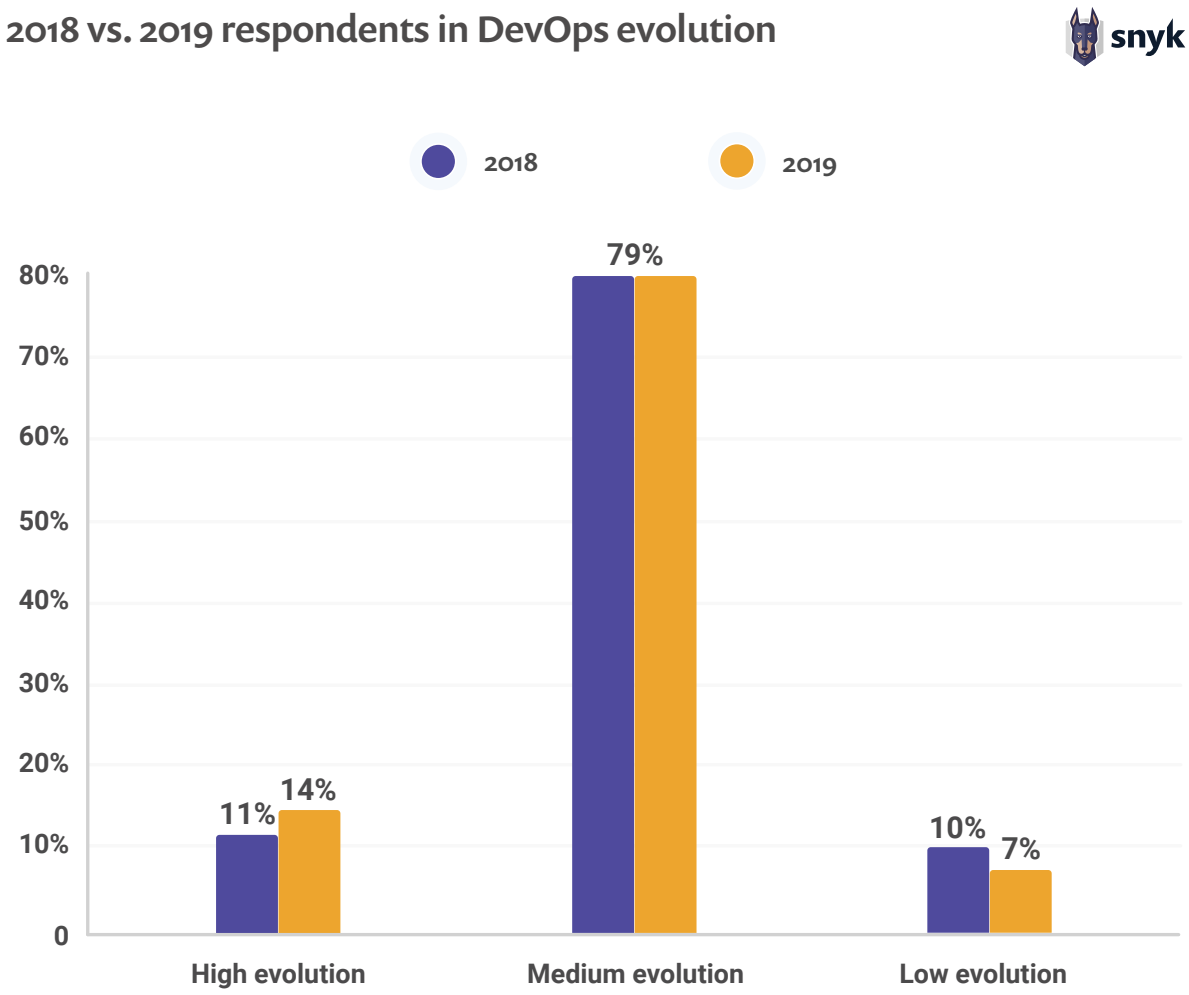
> Only thing is if it is transparent to them [the development teams], they love it, because that's not slowing down the process to production. But if it is slowing down, they want to find better ways to manage this.
>
> *- Mohan Yelnadu,*
> *Head of AppSec at Prudential,*
> *on the Secure Developer Podcast*

State of DevOps adoption

The Puppet State of DevOps report sheds light on the state of DevOps adoption and maturity in different organizations as well as how this impacts security adoption.

One of the findings presented in the report is that 79% of organizations are positioned at the medium-level of DevOps evolution. Organizations also face challenges in the form of scaling tooling, culture, and practices to effectively meet the promises and value of DevOps.

To help with DevOps adoption in the middle stages, Puppet recommends focusing on measuring business outcomes and make use of DevOps-driven metrics. This approach reflects the current status and provides insights into what's to be done next in order to achieve a more mature DevOps adoption.



Based on the Puppet "State of DevOps" report

The DevSecOps promise

There are two crucial assets to adopting DevSecOps successfully:

- Enabling cultural change where security is prioritized instead of being an afterthought — a culture where security is everyone's responsibility and not a siloed ownership of the security team.
- Tooling and automation that empower developer, operations, and security teams to make smart decisions and fix vulnerabilities without slowing down the organization.

Collaboration between dev and security teams at the earliest stages of the software delivery lifecycle promotes a culture of shared responsibility that ultimately provides security teams with the right levels of visibility and control. As Puppet's 2019 State of DevOps Report found, practices like threat modeling have the highest impact on the teams overall confidence in security posture.

Security application testing that is employed throughout the SDLC, is a great enabler for DevSecOps — from developer productivity in an IDE to find and fix vulnerabilities immediately, to continuous integration pipelines of a later stage.

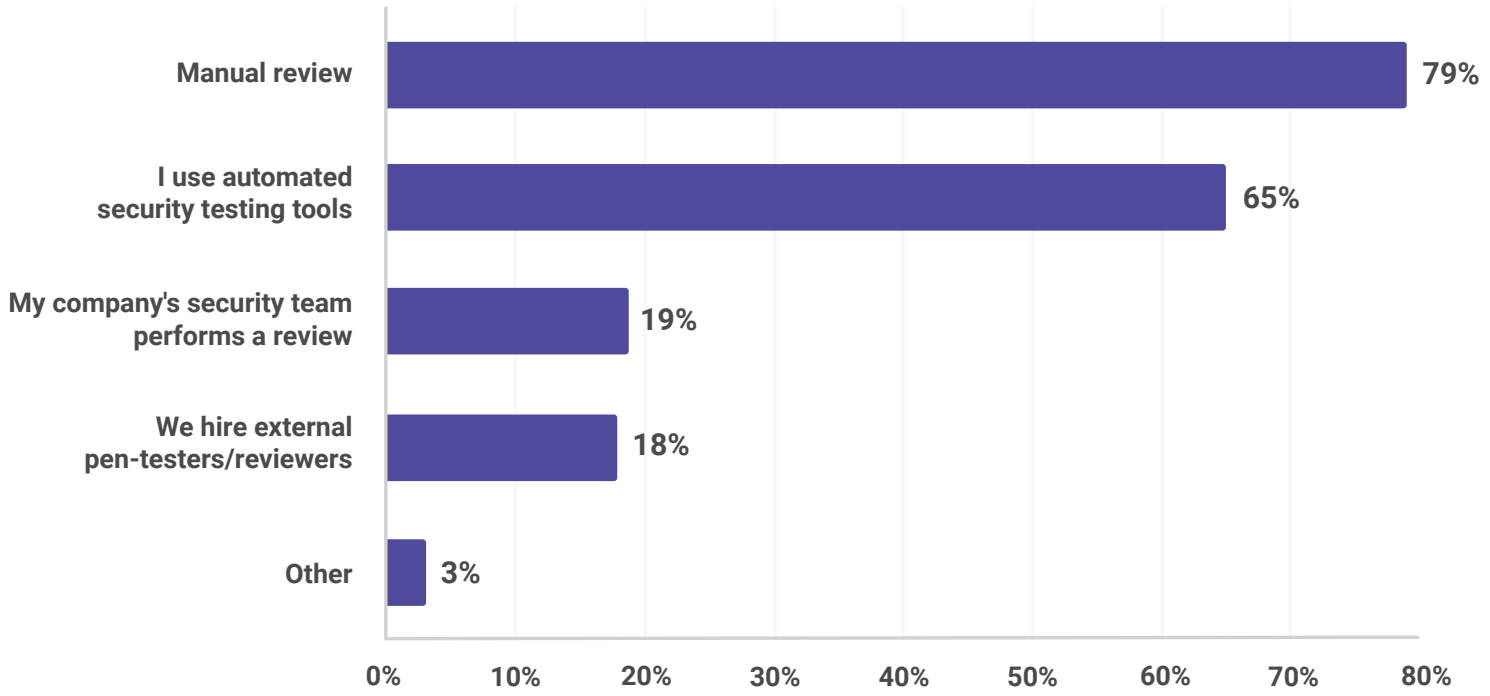
The state of DevSecOps

In this section, we review the state of organizational readiness for DevSecOps adoption, how DevOps maturity impacts security integration, and we go over the lessons learned on security posture of the teams that embraced DevOps.

Organizational readiness for DevSecOps adoption

As we look into the way engineers audit their code bases, we see a strong adoption of automated security tooling, according to the Snyk State of Open Source Security report 2019, with 65% of respondents confirming that observation. It is also important to point out that, even when automated security tools are employed, 79% of the respondents still use security code reviews.

How do you audit your code?



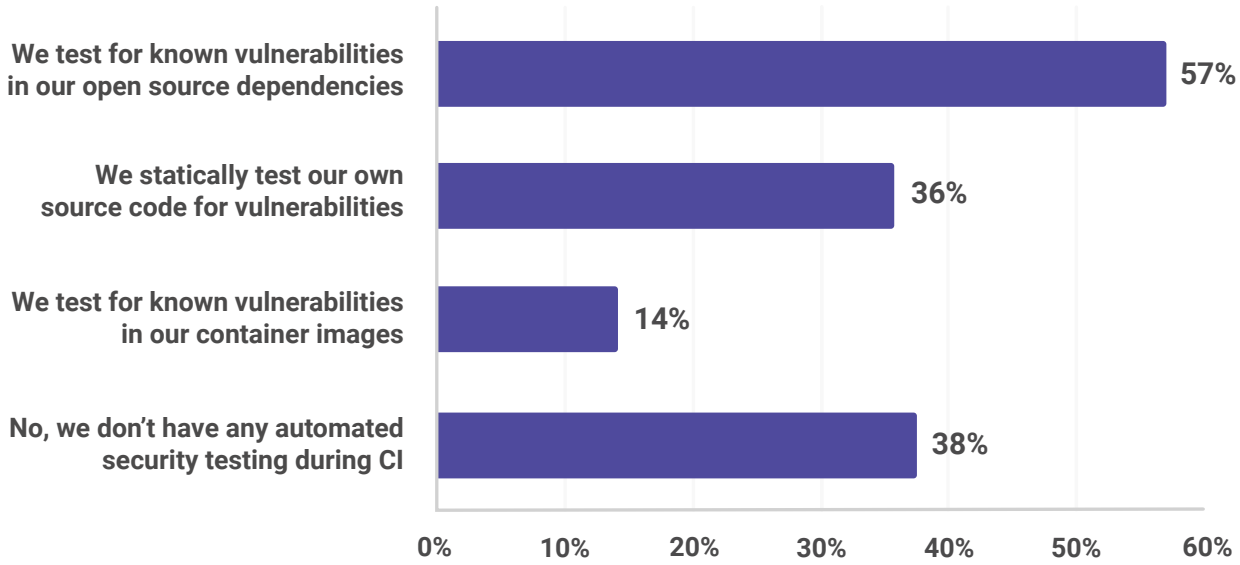
As teams embrace automated security tooling they also recognize that the negative impact the embedment of the tooling in a CI pipeline has on build time, worsens the experience and feedback loop for developers.

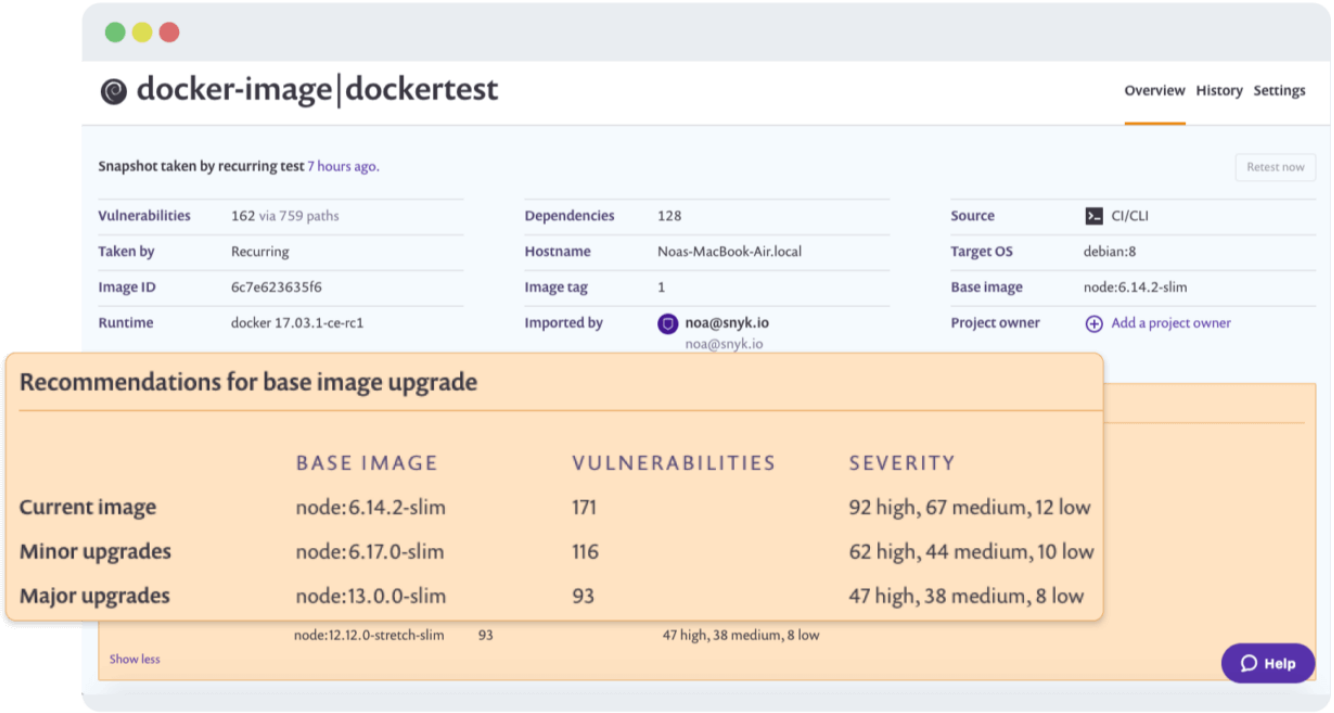
We found that 57% of respondents test for known security vulnerabilities in their open source dependencies while a significantly lower percentage perform static application security testing (SAST).

This is often the result of the latter involving high runtimes for this kind of security testing and also the fact that it results in a high percentage of false positives which then require manual review.

While a little over half of the respondents confirmed they test for known vulnerabilities in their application's open source dependencies, only 14% perform a similar test in container images during a continuous integration pipeline. Is it possible that respondents aren't aware of the security tooling available to them that mitigates this gap? Another option is that with most security tooling, you only get a report of which vulnerabilities exist in the container image, but fixing the actual problem, is entirely up to you.

Do you include automated security testing in your continuous integration pipeline?





As a source of comparison, Snyk Container provides actionable advice in the form of alternative container images that when used, they reduce the number of vulnerabilities, and minimize the overall security exposure.

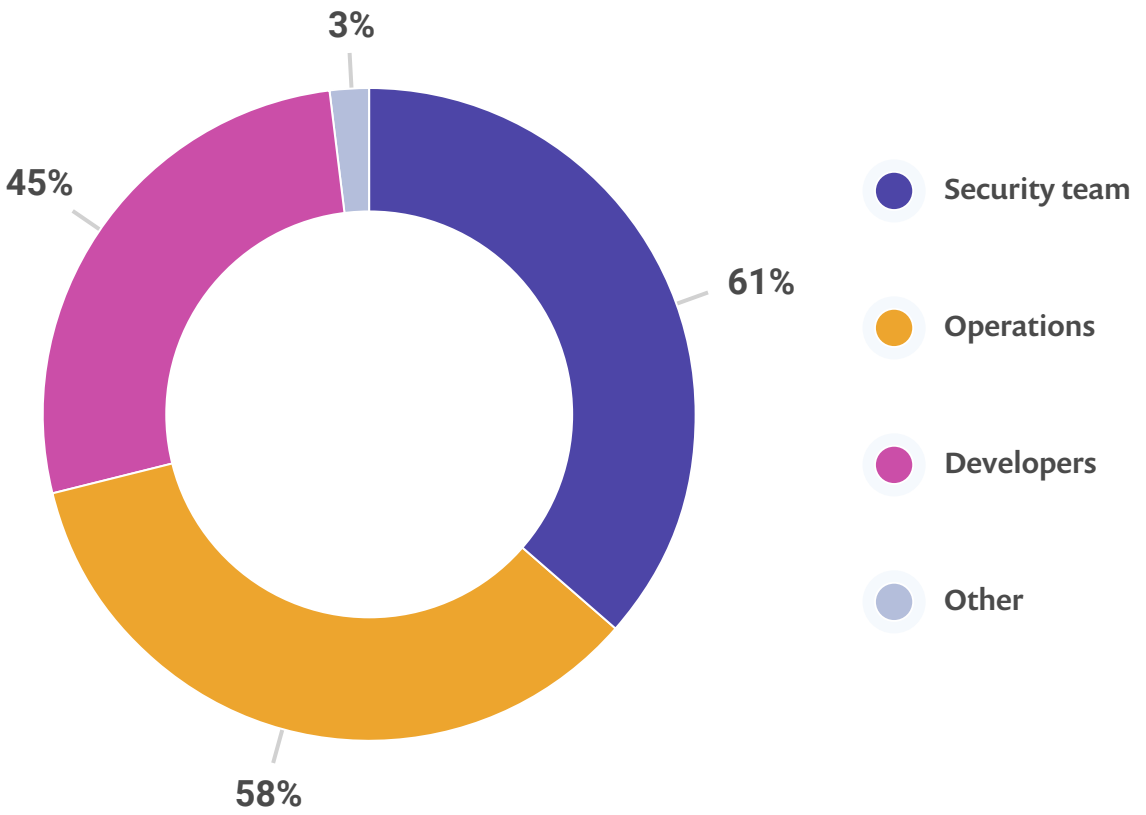
Switching a base image for a docker container is an easily executed action that results in a high return on investment for security. In fact, based on scans performed by Snyk users, the Snyk State of Open Source Security report shows that 44% of docker image scans had known security vulnerabilities for which there were newer and more secure base images available.

Container security expands to more than just Docker container images. It impacts Kuberentes with real security issues in the form of vulnerabilities found in Helm charts. The Snyk 2019 report Uncharted territories: the untold tale of Helm Chart security, revealed several risks in this area:

- 68% of stable Helm Charts contain an image with a high severity vulnerability.
- Updating to the latest published images, reduces the number of vulnerabilities for 64% of the stable Helm Charts.
- 6 images (out of a total of 416) account for half of the vulnerability instances.

When security is the application itself, or its vehicle — for example, container images used to deploy the application — then we found that developers play a key role in owning the responsibility of security for their application. How does this differ when we discuss the responsibility for the security of the infrastructure? Surprisingly, all the parties contributing in a DevSecOps environment, are almost evenly accountable for the responsibility of the infrastructure security.

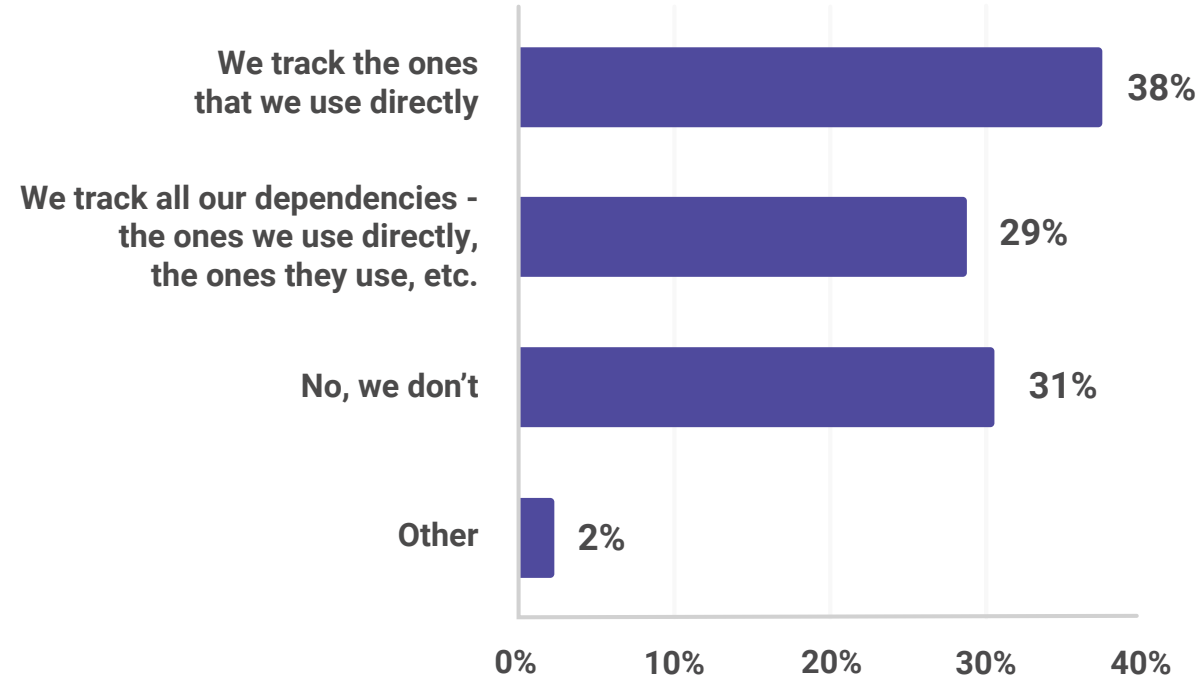
Who do you think should own the security of your infrastructure?



Stepping up for the responsibility of the application's security is a great start. That said, how could one be responsible for their application dependencies if they aren't even tracking it? This is where Software Composition Analysis (SCA) helps.

We found that 31% of respondents aren't performing any tracking of application dependencies used within their organization, while 38% are only tracking direct dependencies. According to the Snyk State of Open Source Security report, in language-based ecosystems — such as Maven Central, RubyGems, and npm — 78% of the security vulnerabilities we found reside within indirect dependencies which could very well be spread further down in the hierarchy, several levels deep. Yet, only 29% of respondents confirmed that they track all of their application's dependencies, whether direct or indirect.

Does your company track which open source libraries your applications are using?



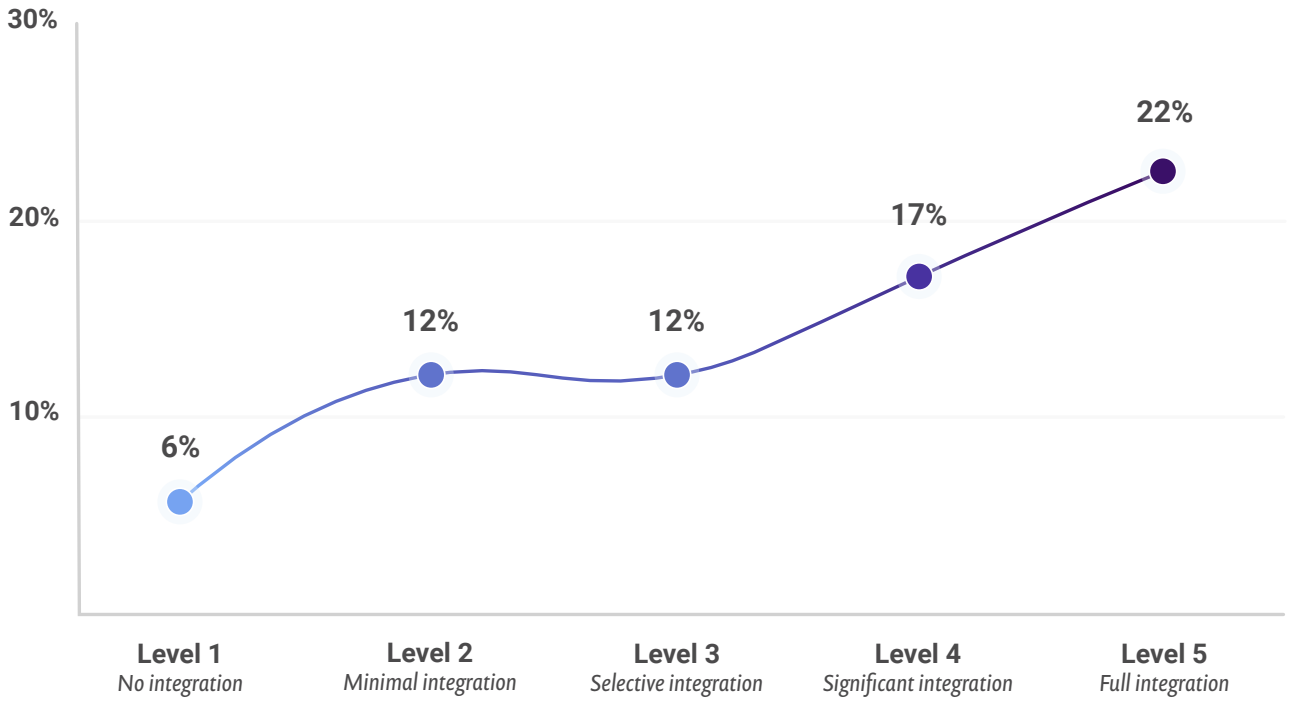
DevOps maturity directly impacts strong security adoption

Puppet's State of DevOps report analyzes how security adoption varies between different organizations depending on their DevOps practices and provides important insights on security posture of businesses.

The more highly evolved organizations are much more more likely to have integrated security across the software delivery lifecycle. The Puppet report finds that 22% of the organizations with the highest level of DevOps maturity (Level 5), are also at the highest level of security integration.

The report also points out that 16% of organizations where at Level 1, the lowest level of security integration. Puppet's findings align well with the Snyk State of Open Source Security report from February 2019.

Percentage of firms at high stage in DevOps evolution

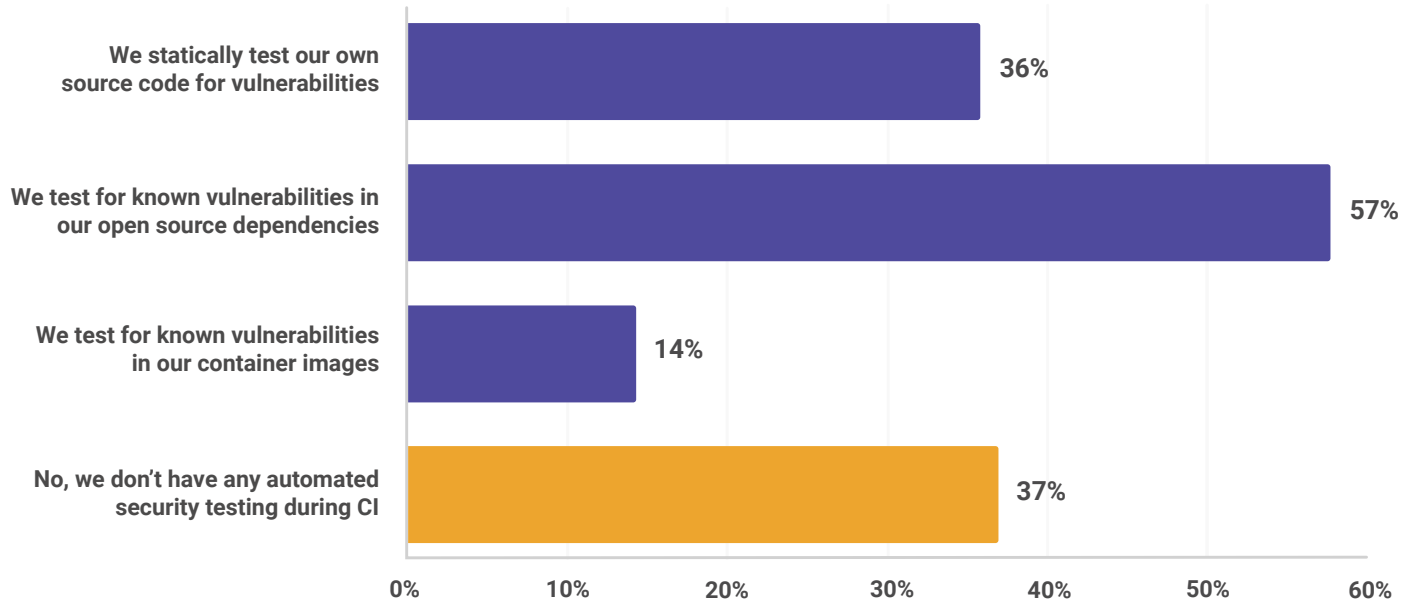


Based on the Puppet "State of DevOps" report

The Snyk report highlighted that a significant 37 percent of the users don't do any sort of automated security testing during a CI phase.

To put this in further context, the Puppet report highlights that the majority (75%) of the organizations at Level 1 of DevOps maturity, get involved with security activities only on an ad-hoc basis, for example, when security issues are escalated from production. This demonstrates that businesses are still at a very early stage of DevOps evolution and maturity. These organizations act reactively to security threats instead of proactively addressing security concerns, not allowing potential hacks and breaches to pose any risk.

Security testing during CI



Lessons learned on security posture and DevOps adoption

The following sections review stories and observations of three key security adoption takeaways.

1 Deep and frictionless security integration benefits security posture and collaboration

A watermark of traditional security activities within organizations is the high tension between security teams, the operations or IT, and the core R&D engineering. When all of these teams are siloed with their activities and overall goals unaligned, they create tension and friction that manifests in mis-executive security activities.

However, when security practices are integrated throughout the SDLC, then the overall confidence level of security practices levels up for the entire organization. Furthermore, the Puppet report shows that, when security activities take place very early in the SDLC, they are more impactful.

More specifically, threat modeling was named as the security activity with the most significant impact on an organization's overall security posture and confidence level. Threat modeling, aims to connect all the business stakeholders — security, development, and operations — and focus on answering some fundamental questions, for example, "what are we building?" and "what can go wrong?". This kind of activity creates a collaborative environment and a platform for open discussion and communication between all Dev, Sec, and Ops parties.

Some of the most influential practices in improving an organization's security posture, according to Puppet are:

- Security tooling that is used within a continuous integration pipeline, aids engineers to stay confident in knowing that they don't introduce known security problems into their codebases.

  Such automated security tools often execute fast, providing a good developer experience with a fast feedback loop that allows developers to move swiftly in a non-blocking manner. Some of these tools also provide actionable remediation that is closely integrated within a developer's workflow. In this case, developers are further empowered to take responsibility over the security of the application they develop.

- Infrastructure-related security policies are reviewed before deployment.

  Infrastructure as code has been an integral part of many DevOps tooling and it's rapidly expanding with cloud native service provisioning and tools such as the Hashicorp's Terraform. Another example of IaC is the use of text-based configuration to provision container orchestration software, for example Kubernetes.

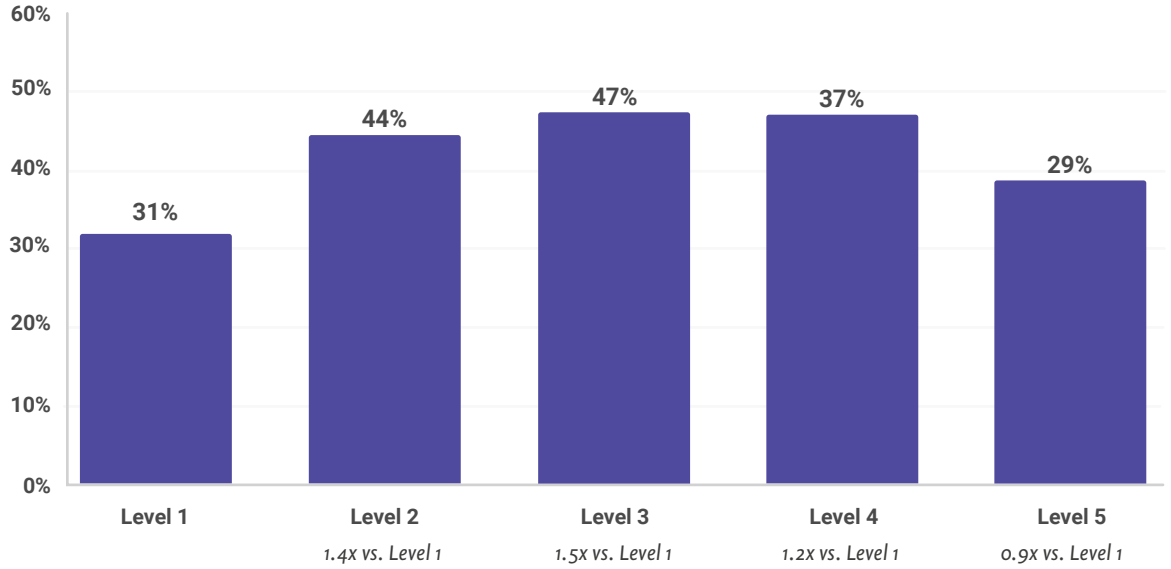
  However, a common security slip up is to accidentally provision an insecure configuration which has a significant impact to an organization. For example, in one particular case, insecure cloud storage configuration allowed improper access to non-authorized users, and resulted in several data leaks, as we highlighted earlier in this report.

That being said, 29% of all organizations positioned at the highest level of security integration, still feel that security teams and delivery teams encounter a lot of friction when collaborating. On the bright side, this is still a better situation compared to 47% of all organizations at the medium phase of security integration who share the same feeling. Notably, when security integration doesn't exist, teams aren't collaborating at all.

Another notable highlight from the Puppet report is that organizations which had a strong and deep security integration were able to prioritize security issues over generic feature delivery, and address them faster. This is a loud statement — when security is viewed as a shared responsibility across the organization, then minimizing security risk to the business takes precedence.

Security integration and friction between teams

Respondents agree security team encounters a lot of friction when collaborating with delivery teams.



Based on the Puppet "State of DevOps" report

2 Deeply integrated security increases the sense of shared responsibility

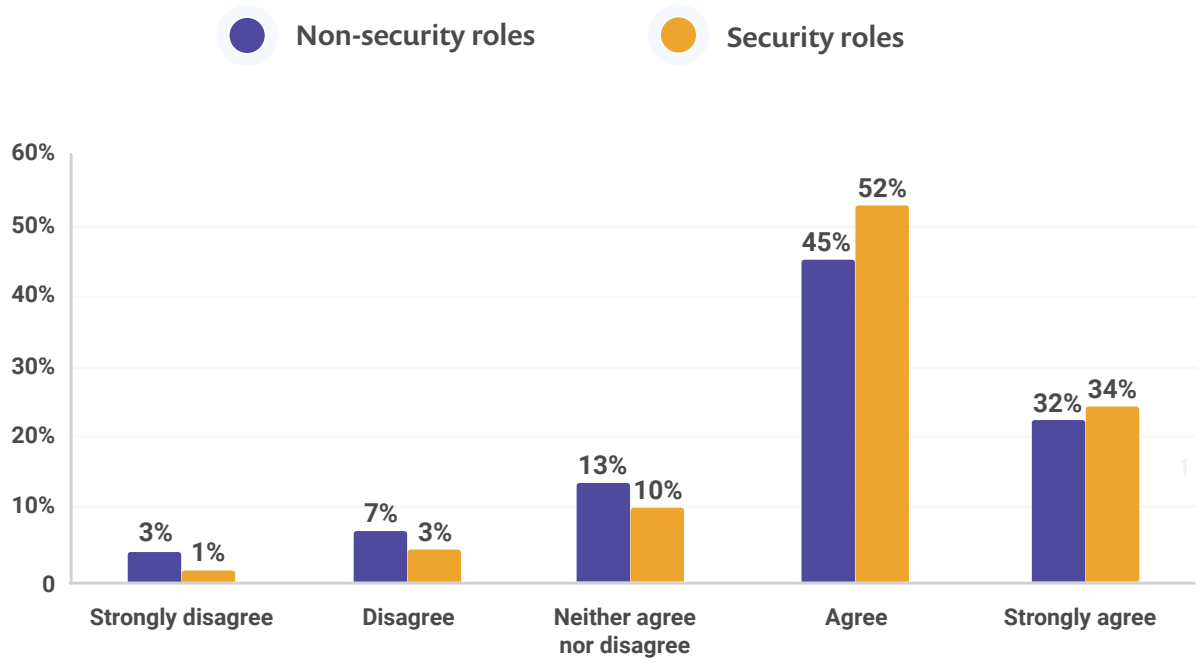
Having a sense of shared responsibility across the organization contributes to an elevated security-first mindset among employees who will seek out to question and challenge solutions regarding the security impact of the products they build.

> We found that the more security is integrated into the software development lifecycle, the more delivery teams see security as a shared responsibility. In fact, seeing security as a shared responsibility improved by 31 percentage points between Level 1 and Level 5.
>
> *- Puppet 2019 State of Devops*

Notably, the report demonstrates that security is a shared responsibility in the place of work, for 86 % of security professionals.

Security as a shared responsibility

Responsibility for security is shared across security and delivery teams.



Based on the Puppet "State of DevOps" report

3 Executing well on DevOps is key to enabling DevSecOps

An organization is more likely to adopt security practices, the higher it lands on the DevOps evolution ladder. When organizations exhibit a strong level of DevOps tooling and culture adoption, they are well positioned to further enable security practices and DevSecOps. In the DevSecOps world, an organization relies on automation as a basis and empowers engineers throughout the organization to collaborate across different departments and take action to improve security.

DevSecOps advice for practitioners, by practitioners

A modern software architecture stack can get complex pretty fast; It is impossible to understand all layers in detail. Therefore, it makes sense to collaborate with other people more versed in other aspects. Similar to ops, DevSecOps adds the security people to that conversation. To have a meaningful dialog, one needs to have a common goal: making the app more secure.

In the past, nobody stopped developers from talking to security people, but in many cases the business priorities favored features and delivery first. More mature organizations that have worked hard on their delivery pipeline, now are ready to take it to the next level. Not just deliver faster, but also improve what is being delivered. The shift left approach empowers developers early in the valuestream, as issues further down the delivery path get more costly. By automating common tasks and creating a platform of collaboration, knowledge can be shared so that awareness increases. While many people thought automation would render us jobless, it actually allows us to improve what gets delivered. Now, the question becomes how much security is enough security? That is, ultimately, a business decision. But now, we can better explain and handle risks that may arise, so the cost/benefit ratio starts making more and more sense, for a business.

- Patrick Debois, DevSecOps Practitioner, author of The DevOps Handbook

For me, DevSecOps is about making security simple and easy for developers. Take least privilege, for example. How many of you inspect IAM roles that are supplied by OSS projects? What about Kubernetes RBAC objects? Do you know what to look for in there?

We can't expect devs to create least privilege roles if it's hard, or if it's not easy to understand what not to do. It's about tooling. But it's also about cultural change.

- Omer Levi Hevroni, DevSecOps Engineer at Soluto.

Many people harp upon the fact that DevOps, if done properly, includes security. The problem with this is that our industry is reliably producing insecure applications, DevOps or not. The term DevSecOps has been coined in order to emphasize the importance of security and make it a real priority for our industry. Although the tools and tactics to create secure applications may change in a DevOps environment, the goals and strategies remain the same; secure requirements, secure design, secure code, all types of testing, performing maintenance, monitoring and patching and having a responsive incident process. DevSecOps is merely a more modern and mature form of Application Security, our industry still strives for the same results; safe, secure and reliable software.

- Tanya Janca, Independent Security Consultant

> A lot of people assume that DevSecOps will replace penetration testing; however, that's not the case. We can not leave the in-depth testing of applications. So, the best way is to create a parallel security pipeline for more in-depth testing.
>
> *- Vandana Verma,*
> *Security leader*

> The velocity at which vulnerabilities can be detected and remediated will be key. It's a race — where organizations are up against a malicious community of individuals and teams of increasing size and professionalism. Their goal is to use vulnerabilities in your software to extract information — data for commercial extortion and national security purposes, and intellectual property for commercial and military advantage. Technologies and processes which enhance development teams' early detection and remediation of security vulnerabilities, together with robust training in secure coding techniques, will help to protect the organization, their clients and stakeholders.
>
> *- Wendy Ng,*
> *DevSecOps Security Managing Advisor at Experian*

snyk

Develop faster. Stay secure.

Twitter: @snyksec

Web: https://snyk.io

Office info

London

1 Mark Square

London EC2A 4EG

Tel Aviv

40 Yavne st., first floor

Boston

WeWork 9th Floor

501 Boylston St

Boston, MA 02116

Report author

Liran Tal (@liran_tal)

Report contributors

Alana Brown (Puppet) (@alannapb)

Eirini-Eleni Papadopoulou (@Esk_Dhg)

Report design

Growth Labs (@GrowthLabsMKTG)