THENEWSTACK

Best of DevSecOps

Trends in Cloud Native Security Practices

The New Stack

Best of DevSecOps: Trends in Cloud Native Security Practices

Alex Williams, Founder & Publisher

Ebook Team:

Emily Omier, Author and Editor

Gabriel H. Dinh, Executive Producer

Judy Williams, Copy Editor

Libby Clark, Editorial & Marketing Director

Matt Chiodi, Contributor

Meera Rao, Contributor

Tal Klein, Contributor

Supporting Team:

Benjamin Ball, Director of Sales and Account Management

Joab Jackson, Managing Editor

Michelle Maher, Editorial Assistant

© 2021 The New Stack. All rights reserved.

20210223

Table of Contents

Sponsor

We are grateful for the support of our ebook sponsor:

Prisma Cloud is the industry's most comprehensive Cloud Native Security Platform (CNSP) with the broadest security and compliance coverage – for applications, data, and the entire cloud native technology stack – throughout the development lifecycle and across multi- and hybrid-cloud environments. Our integrated approach enables security operations and DevOps teams to stay agile, collaborate effectively and accelerate securecloud native application development.

Introduction

DevSecOps is the culture and practice of making security part of the entire software delivery process from start to finish — a discipline increasingly associated with overall resilience and DevOps best practices.

The people who pioneered DevOps were optimizing for deployment velocity. They had to focus on configuring the hardware to fit the software architectures. Until the recent site reliability engineering (SRE) movement, resilience has not been a traditional DevOps metric, just as security was not a traditional DevOps metric until the move towards DevSecOps. DevOps as a cultural movement focused on efficiencies, velocity, feature development and continuous development. The SRE movement is focused on resiliency and the ability to handle spikes in usage or disruptions in cloud provider service without outages or performance hits.

Traditionally, security has sat on the outside, isolated in culture, organization and workflow from development teams, operations teams, DevOps teams and SRE teams. The teams have dealt with different success metrics and different performance indicators. Yet there is clear overlap between DevOps, SRE and security, as application programming interface (API) endpoints become more vulnerable and require better risk management. Which brings us to the new world of DevSecOps.

Modern architectures are forcing deeper integration between DevOps and SRE teams. And the same forced integration is extending to security teams. This is DevSecOps.

Programmatic infrastructure, also known as Infrastructure as Code, binds them together. It provides automation, metrics and analysis to continually develop DevOps, SRE and DevSecOps practices. Metrics overlap across all three disciplines. Customer tickets, for example, are a metric in DevOps. In DevSecOps, customer support security tickets are a metric. There is a mutual interest in decreasing the number of bugs and vulnerabilities across the different disciplines.

For example, vulnerabilities may be decreased by automatically separating the ones that are not loaded into memory and thus don't need to be fixed. Automated vulnerability detection is just one example of how anomalies may be more easily discovered, making the overall infrastructure more resilient to attacks.

Continuous development teams now have to focus on the principles of integrating security from the very start. In terms of resilience, scale-out architectures must consider services as cells that have their own layers of protection but are still connected in loosely coupled environments.

In our first "Best of The New Stack" anthology, we've picked six posts from the past year to provide deeper perspectives about DevSecOps and the trends emerging around it. What you'll find:

- The primary and secondary impacts of DevSecOps.
- Reasons to shift security both left and right.
- Why velocity equals automation.
- How declarative environments are suited for DevSecOps.
- Why treating security holistically allows for better resilience.

So what goes into resilience and reliability when we discuss at-scale application development, deployment and management? And how does it apply to DevSecOps? Observability and chaos engineering come to mind. Both approaches help organizations make their services more resilient and reliable.

Chaos engineering is a concept pioneered by the Netflix Edge tools team. The idea: unleash the chaos monkeys to create havoc for testing the system and in the process, make it more resilient. Observability allows the SRE to find the "unknown unknowns," that uncover issues such as latency problems. Security teams also require ways to separate the signal from the noise as older practices, such as monitoring, which are really meant for legacy architectures, persist. Cloud native architectures, with their container-centric approach, require a different, more

sophisticated way to view across distributed environments, including automated security feedback loops.

Security and resiliency are often interrelated, but the relationship is rarely discussed. Ask even accomplished technologists about security and they will say that security and the software development life cycle (SDLC) are completely separate. More than one engineering executive has made this remark to The New Stack in recent conversations.

DevSecOps has become a necessity as connected container-based architectures depend upon the sharing of data. Each service is integrated into the larger service. Communication in its old-fashioned human form is as essential as automated communications between services.

The more security is built into the SDLC, the more confidence the organization has in its security posture, according to the 2019 State of DevOps Report. And remediation is faster when security is integrated, according to the 2020 State of DevOps Report. Some 45% of organizations that had integrated security into all phases of the SDLC said they could remediate a critical vulnerability within 24 hours, while only 25% of organizations with no integration could do so.

Still, the confusion over how best to integrate security into the SDLC continues, as well as which security tools to use, even among companies with mature DevOps processes. Many practitioners are unsure if they should depend on their cloud service for tools, use third party tools, roll their own, or depend primarily on open source security tools.

Maybe a better way to think about DevSecOps is from the point of view of making security a routine part of an engineer's job. Integrating security into an ordinary development workflow has its challenges but it's a lot better than dealing with crises. An ordinary life is the best life. But achieving an ordinary, crisis-free life requires a lot of work. It means keeping with routines, establishing behavioral best practices and controlling what you can control.

It's a state of tension and anxiety, knowing that security can't be hammered into software development as a last step. That is perhaps how it used to be when tightly coupled architectures were configured, taken down for days and often weeks to be prepared for a new version of the system of record.

In reality, the future is right in front of us and so is the path to true resilience. Container-based architectures are resilient by design. Immutable infrastructure allows for containers to be deleted and replaced with new ones. That's a welcome change from older approaches that did not have that ease of replacement. Programmable security and the accompanying DevOps practices are the next step on this path.

DevSecOps may not be as complex as we make it out to be. It's there already in the form of modern architectures. Now it's just a matter of defining the orchestration itself to automate and scale security practices in environments already resilient by design.

Alex Williams

Founder and Publisher

The New Stack

@alexwilliams

alex@thenewstack.io

SECTION 01

Getting Started with DevSecOps

Author

Emily Omier is a positioning consultant who works with startups to stake out the right position in the cloud native / Kubernetes ecosystem so that end users immediately understand their value. She also hosts The Business of Cloud Native, a podcast about the business reasons that push and pull organizations towards cloud native.

CHAPTER 01

The Motivations Behind DevSecOps

Original article published 26 May, 2020 by Emily Omier, for The New Stack

Think of DevSecOps as an extension of DevOps. Just as developers would, in a waterfall development style, throw projects over the wall to the operations team to make them work in production, DevOps teams would throw projects to security teams, making security totally separate from the development process.

The term DevSecOps is often used in conjunction with "shifting security left." This means the developer of the application is responsible for integrating security into the build and test phase of the application, rather than leaving it as a separate task later for the security team. Most people focus on how DevSecOps integrates security into the development process, especially the continuous integration/continuous delivery (CI/CD) pipeline, but DevSecOps is not just about securing applications pre-production.

"It's an umbrella term. It applies, just like DevOps, both to how we build and ship and deliver software, and how we operate it in runtime," said Guy Podjarny, founder and president of container security company Snyk.

Why Adopt DevSecOps?

Perhaps the relative silence on collaboration between operations and security is because the top benefit most people talk about from DevSecOps is neither improved security nor improved reliability — traditional security and operations metrics. In most cases, the reason organizations seem to adopt DevSecOps has to do with getting applications to market faster — in other words, to meet incentives typically associated with developers.

"I think the organizations that have successfully done DevSecOps, they are able to deliver things to market — new products, new features — much faster than their competitors," explained Matt Chiodi, chief security officer, Prisma Cloud at Palo Alto Networks. "There are pure business benefits."

Kirsten Newcomer, product manager for OpenShift at Red Hat, said that she's seen companies move from a 38-week deployment cycle to a seven-week deployment cycle by adopting DevSecOps.

In a traditional security setup, security teams often don't review code until it's already gone through the entire development pipeline. When problems are discovered, the security team will send the project back to the developers with a list of things to fix. This long feedback loop slows down the project's delivery while also heightening the tension between security and developers.

In practice, keeping security separate also makes security problems more likely to slip into production. "Most organizations spend 80% or more of their time looking at the run phase," Chiodi said. "When security is evenly distributed across build, ship and run, you end up with a much lower level of vulnerabilities that then surface in the runtime."

Nonetheless, among the security professionals interviewed there seemed to be a consensus that improving the organization's security posture is not the core motivation for adopting DevSecOps, but rather a happy secondary effect.

Getting DevSecOps Right

What organization would not want to increase development velocity while also improving security and resilience? Many newer companies adopt a DevSecOps approach from the very beginning, but implementing DevSecOps is not easy for established, large organizations. Moving from DevOps to DevSecOps is probably at least as challenging as moving from waterfall to DevOps.

"The teams I have seen be really successful at DevSecOps have buy-in at the executive level," explained Newcomer. DevSecOps requires a major cultural change, and involves shifting incentives for everyone involved. Without high-level buy-in, that kind of structural change is impossible.

"Anytime you have to change the way people think or act, that's going to be the hardest uphill battle," explained Hillary Benson, head of product at container security company StackRox. "And there's no real established blueprint for how you get to DevSecOps nirvana."

Changing the Security Role

"Security has the reputation of being the team of 'no'," Newcomer said. The goal of DevSecOps is to change that dynamic, so that security is no longer an obstacle on the path to deployment. But that involves changing how security professionals work.

DevSecOps requires a high level of automation. Security teams should not be running manual tests at all. Tests should be integrated in the CI/CD pipeline. Feedback on an application or feature's security profile should be available without intervention from a security professional.

In a DevSecOps model, a security team's role is to create security policies, help developers and operators understand the output from security automation tools, and serve as advisors. Developers do not need to become security experts, but they do need to become more knowledgeable about security and take action based on the feedback from security automation tools. Security professionals also have to become

more familiar with both infrastructure and software development.

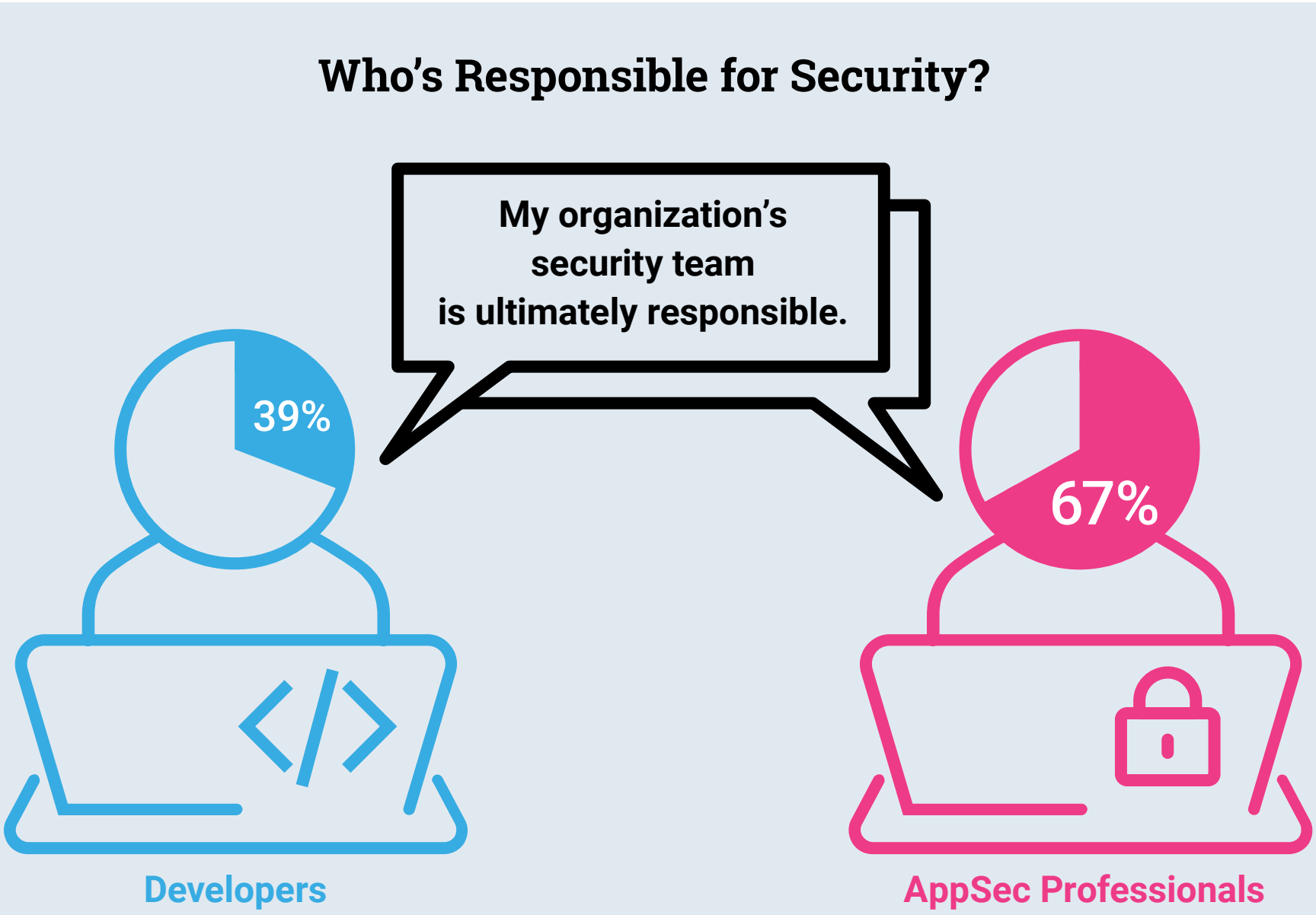
This isn't always an easy path. "If you're a security professional and you've spent a lot of time learning these skills, you're probably not super excited about moving into something that makes you learn a whole different set of skills," Benson explained.

Aligning Incentives

One of the major challenges with DevSecOps is that it involves bringing together teams that are incentivized in different ways. Developers want to get new features out the door fast, security teams want every single potential security risk addressed, operators want super-reliable applications.

This tension is one of the core reasons that a successful DevSecOps transition has to have high-level buy-in. Even with security integrated into the DevOps teams, it's

FIG 1.1: *Two surveys from the Ponemon Institute, sponsored by ZeroNorth in May and June of 2020, reconfirm that although developers have some responsibility for security, the buck stops with the security team.*



Source: Ponemon Institute.
See TNS coverage of the report at https://thenewstack.io/culture-vulnerabilities-and-budget-why-devs-and-appsec-disagree
Icon based on "blogging" by Andrei Yushchenko from the Noun Project

© 2021 THENEWSTACK

important to have shared key performance indicators (KPIs).

So who is ultimately responsible for security? Who gets fired when something goes wrong? That's a sticky question — some people think security should still be the security professional's responsibility, some think that responsibility should shift to developers, others that it should be everyone's responsibility. The risk, of course, of "everyone's responsibility" is that security becomes no one's responsibility.

How Do You Measure Success?

In most cases, development velocity is the key success metric for DevSecOps. There's one big advantage to using development metrics: Security is very hard to put numbers on.

"The difficult part of measuring anything in security, in particular, is that you're sort of measuring the counterfactual," explained Benson. "If nothing gets breached, it means you're doing a good job, or it means you were lucky."

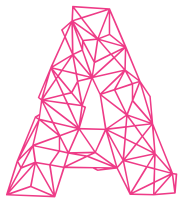
But if DevSecOps is at least as much about cultural transformation as it is about technologies, it might make sense to measure success by organizational markers. For example: Are security professionals spending their time running tests themselves or creating/configuring tools for DevOps teams to use? Are security professionals aware of and taking into account the business rationale for features when they make recommendations? How frequently do development, security and operations talk to each other?

What exactly do developers, security professionals and operators do in a DevSecOps organization? Who is responsible for what and how do roles change? That's the topic of the next chapter.

CHAPTER 02

Roles and Responsibilities in DevSecOps

Original article published 15 Feb, 2021 by Emily Omier, for The New Stack



dopting DevSecOps requires both re-aligning incentives for development, security and operations, as well as changing team members' everyday responsibilities. Here's how developer, security and operations roles and responsibilities need to shift as organizations move towards DevSecOps.

The Security Pro

The role of security professionals in a DevSecOps model is dramatically different from traditional approaches to information security (infosec) and application security. Instead of personally evaluating how secure an update or new application is, security professionals in DevSecOps organizations become internal consultants whose main responsibility is to give DevOps teams the tools and training necessary to deploy secure code and monitor security during runtime.

Security teams' responsibilities in a DevSecOps world generally include creating or selecting tools that developers and operators can use to either catch security issues early in the development process or to respond to security incidents in runtime. They set policies and determine the security strategy. "They are also an escalation point for cases that require deeper expertise," explained Guy Podjarny, founder and president of Snyk. "They should be a governing body so that the organization as a

whole and different teams individually are operating in a way that keeps the organization secure."

"The internal consultants should have a decent understanding of the different pieces in the DevSecOps stack," explained David Widen, director of training at DevOps consulting firm BoxBoat. "They have to figure out how to treat DevSecOps as a product or a service that can be used by all the different teams affected."

The career path for security professionals has also changed. Particularly as security shifts left, it is increasingly important for security professionals to empathize with developers. Traditionally, the path into infosec came through operations (ops); however, an increasing number of security professionals have a background in development. Regardless, being able to understand developers, including what is hard and what is not, is a critical skill for any security team in a DevSecOps organization.

The Developer

According to the DevOps Pulse 2020 report from Logz.io, about 63% of DevOps engineers consider themselves and their teams responsible for security, while only 31% of developers consider themselves responsible for security. DevSecOps does dramatically shift responsibility for security to developers, by including security in the feedback loop that developers are expected to address during the build/test phase.

"Now, a developer does a code check-in and there's some piece of automation that tells the developer sorry, we agreed on this, this piece of code cannot go into a repository, cannot go into production," explained Loris Degioanni, chief technology officer (CTO) and founder of Sysdig. "It's completely impersonal. It's the role of the developer to find this kind of stuff and fix it automatically without a security [person] telling [them] to."

Does this mean more work for the developer? Maybe, because there are more things to address during the build/test phase. On the other hand, there's no release-day

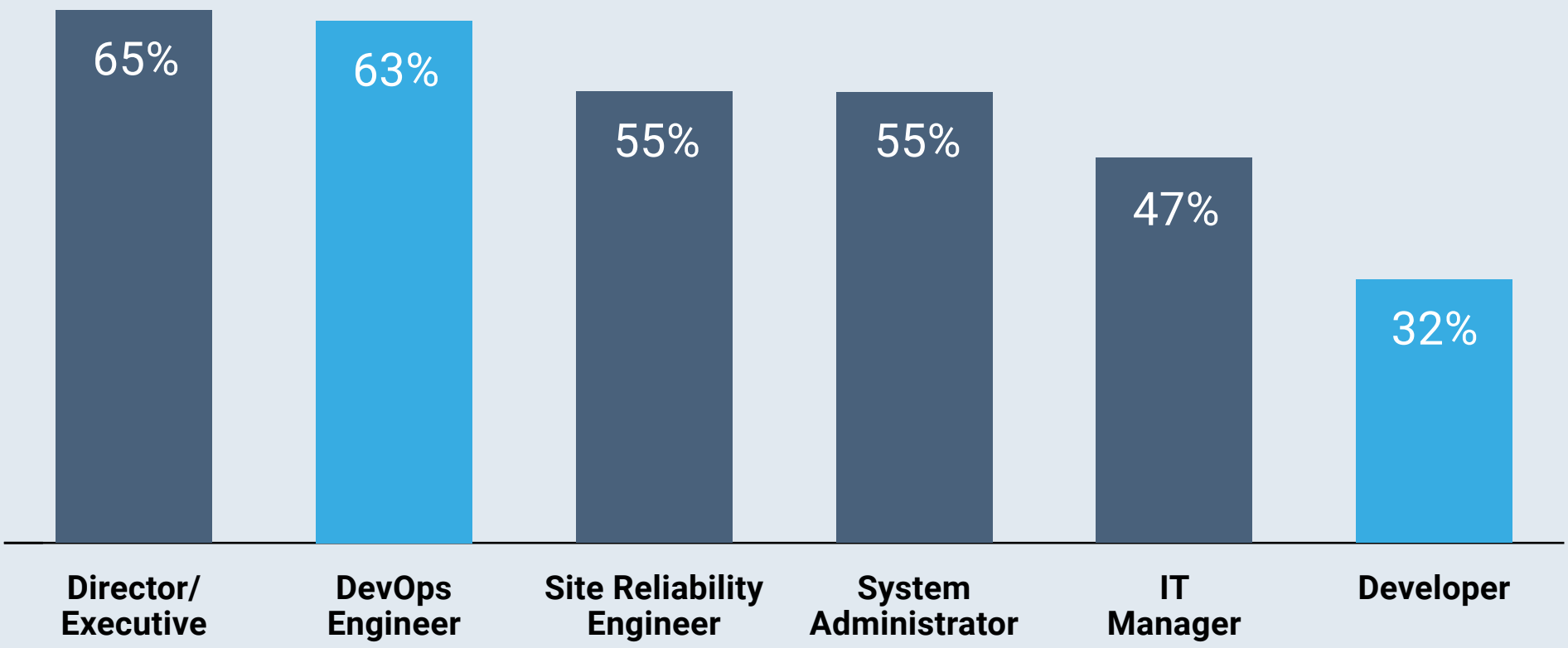


FIG 2.1: *DevOps engineers claim more responsibility for security more often than developers, according to the DevOps Pulse 2020 survey by Logz.io.*

stress when a security professional blocks the release because of a vulnerability the developer didn't know existed. In addition, DevSecOps relies heavily on automation, so the actual manual work that developers have to do to ensure secure code should be minimal. In fact, it's the security team's job to make security as easy as possible for developers.

The SysAdmin and SRE

In the same report from Logz.io, 54% of SysAdmins and 54% of site reliability engineers (SREs) considered themselves and their teams responsible for infrastructure and application security. "Ops were traditionally the ones most aligned with security, and had to apply and operate a lot of security in their day-to-day business," Podjarny said. Indeed, of all the roles in DevSecOps, that of the system administrator (SysAdmin) and SRE are probably changed the least,

because most of the changes in workflow relate to the interaction between code development and security checkpoints rather than infrastructure and runtime security.

New Roles

Adopting DevSecOps also involves creating new roles within your infosec organization. One idea is to create a security champion who can bridge the gap not just between security, development and operations, but also with the executive team to ensure they are on board with security practices, changes in organizational structure and to make sure the security team has the resources it needs to operate.

In terms of other new job titles, there are increasingly organizations looking for security engineers or security automation engineers, highlighting the desire for security professionals who can create and configure custom security tools.

Changing Relationships

As important as the changing roles are, the changing relationships between dev, sec and ops are perhaps even more critical. "It's important to bring DevOps teams, traditional infosec practitioners, infrastructure folks, networking folks, into a single working group or committee," explained Varun Badhwar, senior vice president, Prisma Cloud at Palo Alto Networks. "Really forcing common goals and practices across these teams is the first step in the right direction."

It's also important to understand that any discussion of organizational structure and roles is based on generalities. "If you're a software engineer and you want to get into DevSecOps, you absolutely have to know how security works in a vacuum, but also how does security work specifically in your company?" Widen explained. Understanding the specific context within an organization is critical to everything from setting up appropriate workflows to selecting the right tools. Because everyone runs their technical teams differently and has different priorities and tech stacks, understanding not just the generalities about DevSecOps best practices, but also how

they relate to your specific infrastructure and business, is an essential part of success.

"We're really seeing the conversation shifting to security teams being consultants in the process, really focusing on policy and governance and making sure the requirements are well understood," Badhwar said. "That's a great shift, because it's providing more ownership and accountability to the DevOps teams, allowing them to find security issues earlier in the process and build awareness and training around security."

The people and organizational structure may be the hardest part when it comes to adopting DevSecOps. But tools play a role, too.

CHAPTER 03

The Role of Tools in DevSecOps

Original article published 28 May, 2020 by Emily Omier, for The New Stack

The number one misconception around DevSecOps is that it's just sprinkling in a security tool," said Matt Chiodi, chief security officer, Prisma Cloud at Palo Alto Networks. There is no doubt that the right tools are an essential part of getting DevSecOps right, but they are not enough on their own.

So what role do security tools play in improving development velocity, tightening security feedback loops and integrating security into the entire application life cycle?

Automation, Prioritization, Communication

According to Hillary Benson, head of product at container security company StackRox, organizations need tools to tighten the feedback loop, increase automation and help prioritize which vulnerabilities to fix. "The concept of having a human in the loop every time you need to make a security judgment is just a non-starter these days," she said. Security tools can be integrated in the continuous integration/ continuous delivery (CI/CD) pipeline, and developers can get immediate feedback when a build has a security problem — and the build can be failed and prevented from progressing in the pipeline until the security problem is fixed — all without input from security.

Prioritization is also an important force multiplier. One of the challenges, as companies move to complex distributed microservice architectures, is that there can be a flood of alerts and potential vulnerabilities. Organizations have a limited number of security professionals, and using security tools to prioritize means those professionals can focus on the most important fixes instead of getting bogged down.

"The first thing we do is that we can look at the list of vulnerabilities your scanners produce and reduce it by 50% to 80%," explained Liran Tancman, CEO of cloud native security company Rezilion. "I think 70% of vulnerabilities are never loaded to memory or are not exploitable, so we reduce the number of vulnerabilities that developers have to fix."

> "The concept of having a human in the loop every time you need to make a security judgement is just a non-starter these days."
>
> **— Hillary Benson**

There's a third type of tool that's not specific to security but is nonetheless crucial to success with DevSecOps: communication and collaboration tools. A major component of success with DevSecOps is better knowledge sharing between developers, security and operations. Tools that make that knowledge sharing possible, like Slack or Zoom, are critical to success with DevSecOps — though they are there to facilitate the underlying cultural changes DevSecOps requires.

"A tool will not cause two groups who've never spoken to each other to speak to each other and collaborate," explained Rani Osnat, vice president of strategy and product marketing at cloud native application security company Aqua Security. "It won't make organizational magic happen."

It's the organizational magic that's often the trickiest for security professionals to get right. "When I was at Cisco, vendors would come in and ask 'If I could solve an

issue for you, what would it be?" remembered Rick McElroy, head of security strategy at Carbon Black. "I said, if you can solve these two issues, I'll pay you all my money. It was paperwork and politics. Those are the two things that are always inhibitors for the security program and there's no technical medicine for that."

Living in an Imperfect World

"There was a time when, if there was a memory leak or something that might impact performance, we just wouldn't let [the release] out," explained Tancman. "One of the things that happened with DevOps is we are willing to live in an imperfect world so that we can push code faster. Security is still too often trying to get to a perfect situation instead of figuring out how we live in an imperfect world."

Tools can feel restricting or liberating, depending on both the tool itself and on the larger organizational culture. Security professionals hope DevSecOps will help shift the perceptions of security's role — to become more of a partner than a policeman. Developers tend to take it less personally when a tool fails their build than when a security pro blocks it on release day. The right tools can also build the guardrails, feedback loops and redundancy so that the organization is protected even if developers don't get things perfect.

"How do we bring in the right guardrails that aren't toll gates for security as part of the practice?" asked Derek Weeks, vice president at security automation company Sonatype. Developers want to write better code, and good security tools help them do so. "I think the toll gates are going away and the guardrails are replacing them." Framing it as a way to increase code quality is a way to align incentives between developers and security.

Who Uses the Tools?

"I think that the biggest change that's happened in DevSecOps in recent years is more native DevSecOps tools are being built to serve the developer," Weeks said. In

contrast, in the past, security tools were for security teams.

Since developers are the primary users of security tools, it's becoming increasingly important to make tools that fit into development workflows. "Developers want to be able to stay in their integrated development environments," said Chiodi. "It's important for security tools to speak the developers' language, to integrate with IDEs (integrated development environments) and GitHub, to have native plugins that provide security feedback without forcing the developer to use a different workflow or leave their development environment. This is where the cloud native security platforms and cloud native tooling can really help, because it doesn't require developers to learn new tools or to create a new process."

SECTION 02

Putting DevSecOps into Practice

Contributors

Matt Chiodi has nearly two decades of security leadership experience and is currently the chief security officer for Prisma Cloud at Palo Alto Networks. He is a frequent blogger and speaker at industry events such as RSA. He currently leads the Cloud Threat team which is an elite group of security researchers exclusively focused on public cloud concerns. He also serves as an advisory board member for Rutgers University's Cybersecurity Certificate program and is part of faculty at IANS Research.

Meera Rao is the senior director of product management in the Synopsys Software Integrity Group. She focuses on helping clients establish and enhance their DevOps and CI/CD lifecycles, by integrating a broad range of security-related activities. She has over 20 years of experience in software development organizations and has overseen and performed secure code reviews, static analysis implementations, architectural risk analyses, secure design reviews and threat models of systems.

Tal Klein is CMO at Rezilion, the industry-leading autonomous cloud workload protection platform. He has more than 20 years of experience in the IT and information security industry — working with leaders and exciting emerging vendors in cloud security, client virtualization and networking and data communications.

Editor's Note

Understanding the theory behind DevSecOps is not hard — but actually putting it into practice is. We've assembled some of the best articles about how to implement DevSecOps, how the same tools and practices that increase applications' resilience can also be used to improve security and where DevSecOps is headed in the future.

Secure code is good code, just like resilient, reliable code is good code. Acknowledging that helps get development, operations and security teams on the same page, working together to ensure that every piece of code shipped to production is as secure, resilient and reliable as possible. As the authors in the following articles highlight, Infrastructure as Code and Policy as Code can both be used to improve not only application resilience, but also to prevent drift in security posture during runtime. This is a trend we expect to accelerate in the next few years as cloud infrastructure itself is abstracted into managed Kubernetes services and the traditional DevOps practices of provisioning and deployment shift up the stack and to the left.

No security posture is perfect, either, in terms of processes, tools or the code itself. Just as DevOps teams should prioritize constant iteration to improve everything from reliability to user experience, DevSecOps teams should continually evaluate how the processes, organizational structure, tooling and security guardrails contribute to more secure, more resilient applications that are delivered faster than ever.

CHAPTER 04

5 Steps to Implement DevSecOps

Original article published 23 Jan, 2020 by Matt Chiodi, Prisma Cloud by Palo Alto Networks

The 1980s gave us many good things, such as U2, Metallica and Bon Jovi (questionable). But from a security perspective, this hair-band era is where the proliferation of security tools began.

Fast forward to today and it seems the IT industry's rush to invest in security tools was not a good idea. We also haven't learned from our mistakes. Although research has shown that 65% of cloud incidents were the result of customer misconfigurations, there is still a knee-jerk reaction, with every major breach, to buy yet another tool to fix that particular issue. This is problematic for two reasons:

1. Security teams are not growing proportionally to the tools they purchase. And ...
2. Cryptographer Bruce Schneier was right: Security tools don't make us more secure, processes do.

Given that there are many security requirements across every cloud technology, security teams must focus on streamlining their security portfolios. How can we avoid the sins of the past? The answer lies not in yet another seemingly sexy point product, but rather DevSecOps as a well-defined process.

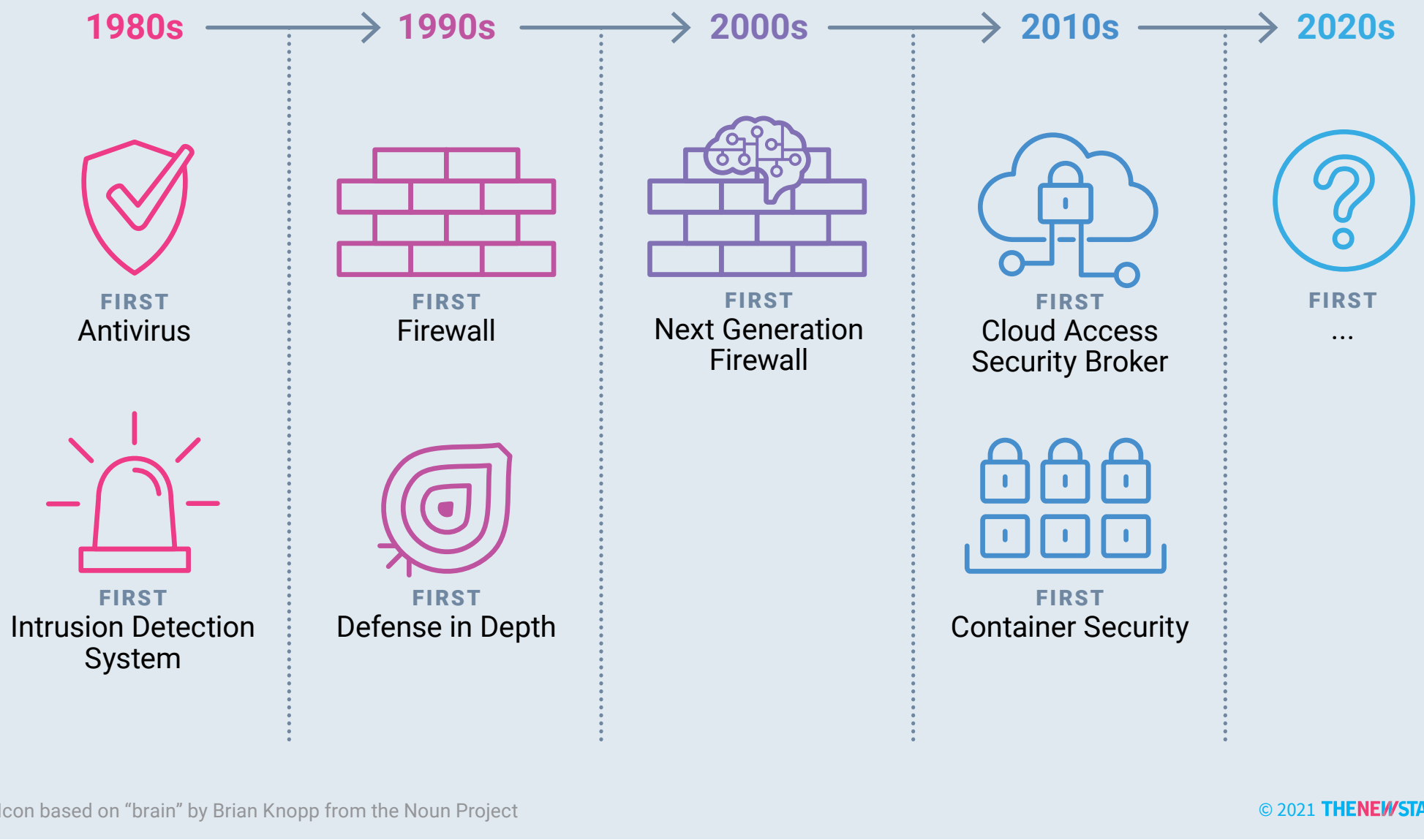


FIG 4.1: *The next generation of cybersecurity tools will be built to support continuous improvement and a DevSecOps strategy.*

Step 1: Define Your Future

Before jumping into this project, it is absolutely imperative to know exactly where you want to end up. If you as the security leader cannot clearly define what the end result should look like, your team will struggle. This isn’t about the technical details of how or the method by which it gets done (this is why you have a team) but rather the outcome you want to achieve. Key items include a few statements on what success looks like, accountability, responsibility, resources and milestones. Expect your strategy to mature over time and don’t spend too much time trying to make it “perfect.” Iteration over time is a key component of a DevSecOps mindset.

Step 2: Discover Code Movement

Whether the security and IT teams know it or not, every organization has a process by which code and changes make their way into the cloud, public or private. The trick

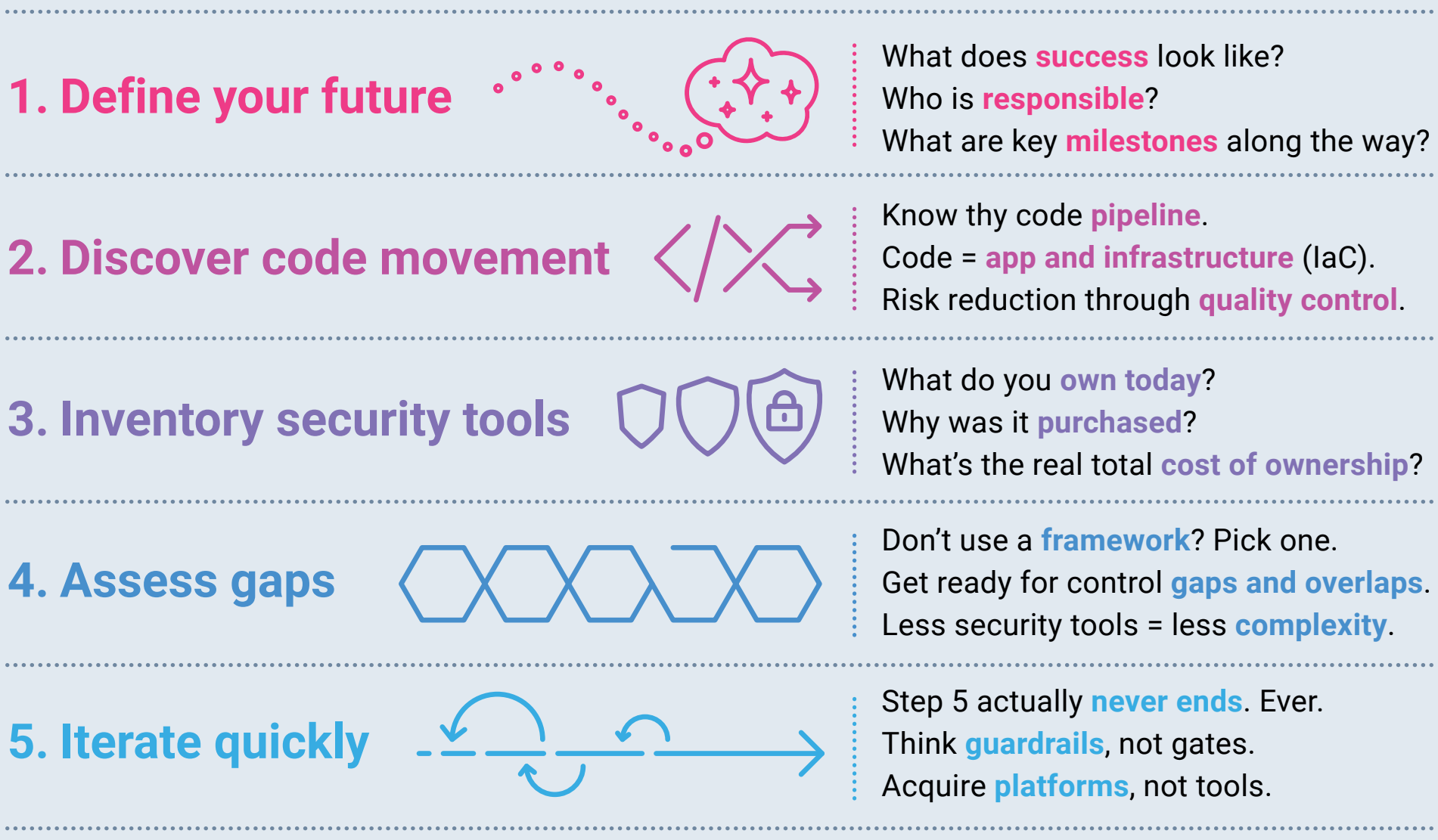


FIG 4.2: *Tool selection should change over time to help execute a DevSecOps strategy that works for an organization's structure, workflows and needs.*

for the security team is discovering what the process looks like today. This is about mapping out the who, what, when and where of how your organization pushes code (application and infrastructure) into the cloud. If this is not well-defined in your organization, then it is highly likely that focusing here may yield the greatest opportunity for improvement, i.e., risk reduction through quality control.

Step 3: Inventory Security Tools

While it's tempting to think your organization can jump to a DevSecOps model, it is not possible without first understanding what is already in your security portfolio. When I ask security teams if they have a list of all security tools in use, a vast majority of the time the answer is no. This isn't surprising as the size of the organization has historically been proportional to the number of security tools. From what I've seen, small businesses can have as few as 20 tools while the largest

of organizations often have more than 130 (think financial services). In this step, your team will create an inventory of all existing tools: commercial, homegrown and open source. Beyond just a list of tools it is important to track, at a minimum, the following key items:

1. Why the tool was originally purchased or created.
2. The risk(s) it was purported to reduce or mitigate.
3. Its native ability to consume and integrate with cloud provider application programming interfaces (APIs).
4. The openness of its API (how easy is it to get data out of the tool).
5. Its ability to generate and share contextual threat intelligence (related to #4).
6. Annual cost (be sure to include hard dollars paid to the vendor as well as an approximate estimate of the cost to support the tool with personnel).

Step 4: Assess Gaps

Many organizations use control frameworks such as the Center for Internet Security's Top 20 Critical Security Controls, NIST's Cybersecurity Framework or the Australian Cyber Security Centre's Essential Eight. If your organization uses one of these, or perhaps relies instead on a risk-based framework, this next step involves overlaying this information with your inventory of tools, as well as the code movement patterns discovered in step two.

However, no matter which framework your organization uses, it is important to base your gap analysis on an industry standard. This analysis should yield multiple outcomes. First, it will help you understand which tools you own, manage and pay for today. Second, and most importantly, it will give you a direct line of sight into both control gaps as well as overlaps. And finally, it will help you identify how you are invested across the security vendor landscape.

As the leaders in this space have consolidated point products into comprehensive

SUMMARY

DevSecOps, aka "shift left" security, integrates security into all stages of the software delivery process, from developers writing code, to testing before deployment, to quick remediation by IT teams.

KEY PRODUCT

Prisma Cloud delivers the broadest security and compliance coverage — for apps, data and the entire cloud native tech stack — throughout the dev life cycle and across multi- and hybrid-cloud environments.

KEY PARTNERS

Amazon, Google, HashiCorp, Microsoft and Red Hat

KEY ACQUISITIONS

Twistlock

cloud security platforms, organizations can save real dollars. They can also use these platforms to reduce the complexity that improved operational efficiencies offer. Critical to achieving DevSecOps is moving from a tangled web of disjointed solutions to comprehensive platforms that support the execution of your chosen framework.

Step 5: Iterate Quickly

The "final" step of the process (okay, it's not actually final, as DevSecOps requires constant iteration) has two distinct parts. The first is taking what you learned in the gap analysis and applying it to your code pipeline, while the second is investigating and acquiring platform-based cloud security controls that support the execution of your DevSecOps strategy. It is likely that this analysis will mean saying goodbye to many of the point products that have overburdened your team for years. Key outcomes for this step include working closely with development and IT to insert security processes and platforms into the least disruptive areas of your code pipeline. This is done effectively through the rapid addition of security guardrails (not gates) along the way.

Taking a continuous improvement-driven approach, fueled by an industry standards-driven gap analysis, will generate ample opportunities for improvement. All without requiring 99-plus security tools. Teams following this process will be well on their way to implementing DevSecOps. Start small. Ramp quickly. Iterate continuously.

CHAPTER 05

DevSecOps is Having Its 'OK Boomer' Moment

Original article published 22 Jun, 2020 by Tal Klein, Rezilion

DevOps veterans know that legacy approaches to software compliance and security do more harm than good, often involving preventive controls that are time-consuming and require manual processes and workflows. Things like access policies, procedures, standards and network firewalls are antiquated; they were designed for waterfall development methodologies and relied on long time cycles, which are incompatible with DevOps.

The debate these days is over how to apply the cattle versus pets principles of DevOps — that enable immutable infrastructure — to compliance and security. I've been a big advocate of Gartner's recommendations for using its continuous adaptive risk and trust assessment (CARTA) methodology for supporting DevOps, because in cloud workloads risk is fluid, not static. It needs to be discovered, continuously assessed and mitigated.

What is Your Desired State?

Here's the elephant in the security room: Resilience is baked into every immutable application or service. That is, in the event of a component or code update, if your best practice is to create a new container with the updated version of your application and delete the old one, then you will have a fresh instance after each

update. Should a vulnerability exit or an injection be attempted, they will be cleaned during the update.

For most, achieving an operational model in which no configuration changes, patches or software updates are allowed on production systems is not quite a reality yet. There will be a day where all patches and updates are applied to "golden" images and layers, and then production workloads will universally be built fresh from these images and replaced, rather than serviced. In such a world of immutable infrastructure, cloud workload protection strategies will shift to a focus on application control and desired state enforcement at runtime.

Today, we're already seeing the world of patch and vulnerability management fall into chaos because the rate of code change and dependency on third-party or open source components is greater than most security and compliance teams can keep up with.

Desired state enforcement solutions analyze the continuous integration/ continuous delivery (CI/CD) pipeline and turn code into a whitelist, establishing an automatic policy that enforces developer intent or desired state. The key with desired state enforcement is to rely on the declarative nature of code to drive policy, rather than attempting to establish a heuristic analysis or attempt to learn good from bad using machine learning — because those methodologies can't keep up with the rate of change and pace of scale in modern DevOps environments. Baselines are a thing of the past.

The declarative nature of modern development languages, modern frameworks and modern architectures inherently creates a whitelist for each service. A recipe or cookbook is basically a policy — it's an architecture policy that tells code what it can and can't do. Why shouldn't we apply the same principles to protection? Cookbooks and recipes are a perfect profile of the desired state — why not use those to populate and build a whitelist, or if you prefer a policy that is enforced at runtime, using your existing immutability mechanisms?

Cloud Workload Protection is Already in Your Code

There are two fundamental benefits of cloud workload protection:

1. Desired State Governance

Understanding what should and shouldn't be running in production environments helps create a better model for architectural hygiene. Containers ship with lots of unnecessary glut that often gets deployed in production. This "inherited" code often suffers from bloat (more code than is needed or is useful). Old code tends to stay around because it's hard to tell if anything or anyone needs it. Safer to leave it in, right? In addition to the problems of identifying who wrote it and when, bloated services have a direct security impact because more code equals greater attack surface.

Older code usually has vulnerabilities, in part due to the evolving nature of the security environment, and new attacks targeting old code appear on a regular basis. Also, the security context of old code can change as applications evolve or code gets moved. Because declarative architecture can be unraveled into relationships, dependencies and actions, desired state enforcement can reliably identify which artifacts are actually needed in production and which components are merely operational bloat.

2. Desired State Enforcement

In today's development environment, we can take advantage of the declarative nature of modern development languages, modern frameworks, modern architectures and cloud native architectures based on microservices, application programming interfaces (APIs) and containers. There's a lot of declarative information there. We can take all of that declarative intent and use it to build the whitelist that we are talking about — it can be our policy.

Let developers do their thing. The onus on the security team is to integrate

protection into the developers' world — rather than complicate it. As Neil MacDonald, distinguished vice president analyst at Gartner, says, "We're not going to go ask the developer to go to some security console or to go write a manifest of all of the applications that are supposed to be on this server. They want to write code, they want to do it quickly and they want to get it into the hands of your customers. We can't slow them down, and that needs to be a guiding principle."

CHAPTER 06

Intelligent Orchestration: The Key to the Future of DevSecOps

Original article published 1 Dec, 2020 by Meera Rao, Synopsys

Modern software development embraces the concept of more: more code, in more languages, on more platforms, with more deployment options. And the natural reaction to this increase in, well, everything, has been the rise of the culture of DevOps in organizations. But DevOps isn't anything new to development and operations teams — they've been using DevOps practices to build high-quality software quickly for years.

DevOps depends on automation to maximize velocity and continuous improvement throughout process feedback. Yet the greatest threats to any DevOps environment are the threats you don't yet know about.

If you're forewarned, you can be forearmed.

Although it's true that DevOps environments enable faster deployments to production, they also pose great threats to an organization if not properly secured. Attackers are on the lookout for DevOps environments, because they know they may offer access to source code, libraries, cloud environments, defect reports and more.

In reaction to these threats, security teams are increasingly adopting DevOps

methodologies to infuse security into already mature DevOps practices. After all, more code and more complexity means more places where things can go wrong.

More velocity also means less time to get things right.

Seamlessly Infusing Security into DevOps

Traditional security tools often cause friction, decrease velocity and require time-consuming manual processes. Very un-DevOps.

Such legacy application security approaches limit a security team's ability to deliver timely and actionable security results, along with the instant feedback needed to drive improvements at the pace of modern development. The result is a continued increase in security threats and vulnerabilities, despite a growing awareness and interest in application security.

In an effort to bridge this gap between DevOps and security teams, innovative new solutions are emerging to address challenges — such as finding the best way to approach security holistically, from a boots-on-the-ground perspective around risk reduction.

The key? Better automation.

Rather than running full scans every time you make a change to the code, there are now solutions coming to market that use intelligent test execution — based on context — to decide what test to run, when to run it and how to run it. This frees teams to handle each application with limited configuration and the flexibility to easily adjust policy in one place.

Critical Functions of Security in DevSecOps

Speaking of policy, let's take a moment to discuss Policy as Code. Policy should be based on an informed approach to risk education. A simple yet ineffective policy for software security would be to require all application development teams to use static analysis. A better policy would specify the tool to be used, and identify the

configuration and types of results required before releasing an application.

One way to articulate policy is in the form of machine-readable files — that is, Policy as Code. Using Policy as Code means that the policy is precisely specified, and changes made to it are readily understood and supported by a change management process. This also implies there is something that knows how to interpret and apply the policy. Rather than spreading interpretation across multiple tool integrations, you can use a single integration layer to provide security to your usual processes.

This security layer in DevSecOps performs several crucial functions. It contains and enforces the Policy as Code — what testing should be done, what kinds of results are allowed (or will break the build or deployment), what kinds of findings will be sent to the regular issue-tracking system, what kinds of compliance activities need to happen, etc. Codifying this makes it unambiguous. Having it in one place makes your team nimble when responding to change. Multiple policies can exist, each reflecting a different kind of application and different risk profile.

It also performs appropriate testing at specified events within the development pipeline, as dictated by — you guessed it —the policy, but optimized for the current state of the project. The security layer should also handle tooling integrations and normalize the results from the security tools. Directed by the policy, it feeds findings into your issue-tracking system, where they can be handled like any other issues.

When events such as a repository commit or repository merge request occur, your pipeline asks the security layer to perform security testing. The name of this powerful process is intelligent orchestration. It consults the policy to understand what testing is appropriate, and then — based on the policy, the testing that has already been conducted and some common sense around what has actually changed — the security layer optimizes how the testing takes place.

For example, if a developer has just committed a change to a cascading style sheet (CSS) file, intelligent orchestration determines that a full static analysis scan and a full software composition analysis (SCA) scan aren't necessary. For changes to a few

Java source files in a specific module, incremental static application security testing can be performed to optimize speed.

The Future Is Bright

Integration is an order of magnitude easier from your existing pipeline. This security layer is designed to simplify integration. If you're trying to build a DevSecOps strategy, applying the tools and processes you need might be challenging. Treating security holistically as its own layer can help ease your integrations and make your process resilient to future innovations in the field.

Whether someone discovers a new way to carry out security testing, or you just want to incorporate an additional type of testing or tool within a security layer, the integrations in your development process don't have to change at all. You just need to integrate from the security layer to the new tool and perhaps adjust your policies.

And there you have it — the Sec earns its place in DevSecOps through intelligent orchestration.

Conclusion

The lessons from the stories in The New Stack's Best of DevSecOps anthology relate to the need for resilience to create great user experiences without compromising security. Working backwards, behind the interface is a networked infrastructure that requires multiple teams to work together, according to policies that allow people to perform their tasks without too many surprises. It's the policy that matters. Without it, resilience is eroded.

We might all strive for perfection, but it will always remain elusive, in life, in software development and certainly in security. Failure is an opportunity to learn and improve the system, to make it more resilient over time.

The answer? A cultural shift, really. There's no longer a quality assurance person to double check code quality, and there is no longer a security person to double check for vulnerabilities. Developers need to trust the automated controls, the Infrastructure as Code and the policies they enforce.

There's one problem, though, that will stop organizations from adopting immutable infrastructure. It's risk. Rezilion's Tal Klein said it well: "Security teams need to develop a higher tolerance for risk."

Get beyond the fear and security teams will find themselves catalysts for immutability, allowing organizations to reap the benefits from programmable infrastructure and build increasingly resilient systems and services.

- Alex

Disclosure

The following companies are sponsors of The New Stack:

Accurics, AppDynamics, Aspen Mesh, Amazon Web Services (AWS), Bridgecrew, Capital One, Check Point, CircleCI, Citrix, Cloud Foundry Foundation, CloudBees, Cloud Native Computing Foundation (CNCF), Cockroach Labs, Codefresh, Confluent, Cox Edge Services, Cribl, DataStax, Dell Technologies, Dynatrace, eBay, Equinix, Fauna, Futurewei, GitLab, Gremlin, HAProxy, Harness, HashiCorp, HPE, InfluxData, Infoblox, Intel, Kasten, LaunchDarkly, Lightbend, Lightstep, LogDNA, MayaData, MinIO, Mirantis, MongoDB, New Relic, Okta, PagerDuty, Puppet, Pure Storage, Red Hat, Redis Labs, Rezilion, RudderStack, Sentry, StackPulse, StepZen, Styra, Synopsys, Teleport, Tetrate, The Linux Foundation, ThousandEyes, Thundra, Tricentis, TriggerMesh, Vates, VMware and WSO2.