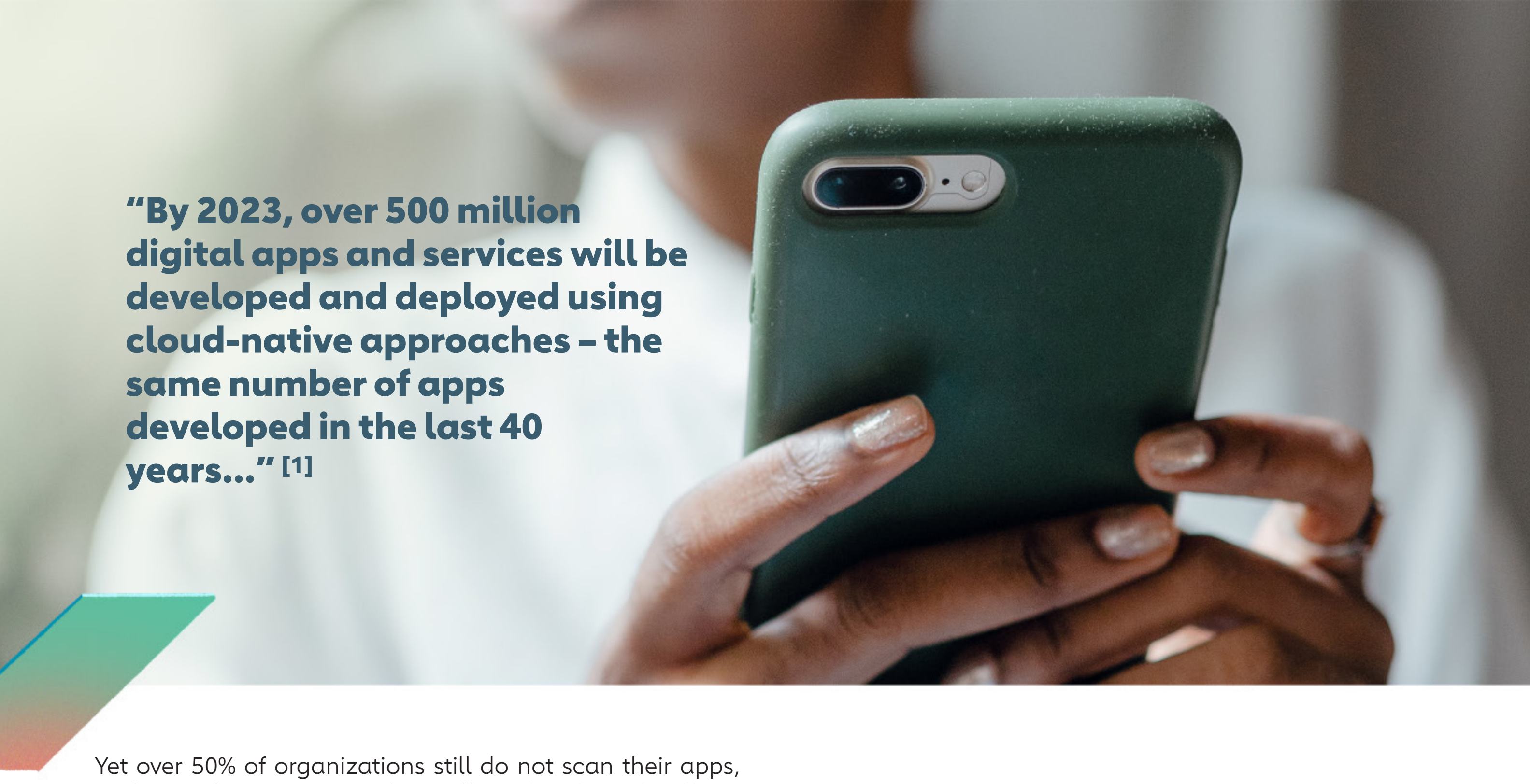
Putting the Sec in DevSecOps

GUARDRAILS

www.guardrails.io

Contents



"By 2023, over 500 million digital apps and services will be developed and deployed using cloud-native approaches – the same number of apps developed in the last 40 years..." [1]

Yet over 50% of organizations still do not scan their apps, and 66% of organizations that suffer a data breach will not survive. For those that do, expect a loss of market share. Indeed, over a 3-year period, the average share price of 34 NYSE listed companies, each of whom leaked at least 1 million customer records when hacked, was down by -15.6%[2]. However, given that between 23% and 53% of customers will jump ship due to a loss of trust, other long-term consequences, such as reputational damage and loss of competitive advantage, remain unknown.

Unfortunately, those organizations still clinging to old, inadequate, and risky security measures are playing Russian Roulette. Their DevOps has been streamlined and working well for some years. However, they still haven't achieved Shift-left—integrating security testing throughout their entire Software Development Life Cycle (SDLC)—and, consequently, are way short of attaining DevSecOps. Unfortunately, time is running out, and that DevOps must become DevSecOps is no longer optional. Any security practitioner knows that security must become integral to their business, that their entire security approach must change, and that it must happen sooner rather than later. With threats...

"...organizations need to plan according to the mindset of, 'It's going to happen to us; it's just a matter of when.' " [3]

With DevSecOps, the old saying, "If it's worth doing, it's worth doing well," springs to mind. As discussed in the *How to build a DevSecOps* pipeline white paper, to be successful, DevSecOps must be approached diligently, carefully planned, and implemented correctly. Unfortunately, all this takes time, and you will be vulnerable if you lack effective Application Security (AppSec) throughout.

Doing nothing is non-viable, and there are typically two approaches. First, you can approach DevSecOps diligently, plan carefully, take it slowly, and hope and pray that attackers are busy elsewhere while you get your house in order. Second, and ever conscious of the urgency, you can cut corners and rush the implementation. We wouldn't advise the latter, as the lack of diligence results in hidden security vulnerabilities that only come to light when exposed by attackers (and by then, it's too late). Your AppSec needs to be in place, and you must be secure from the outset. Unfortunately, there has never been an easy, quick, or effective way to implement security into DevOps. Until now.

Today, as we will show in this *Putting the Sec In DevSecOps* white paper, you can have a fully-fledged DevSecOps pipeline that covers all current and future repositories installed and working within seconds. A pipeline that gives you that much-needed peace of mind and breathing space to concentrate on planning, testing, and implementing the rest of DevSecOps. But that's not all. This solution delivers other significant benefits, including eliminating your security bottleneck and increasing your speed to market, allowing you to ship better, more feature-rich, and more secure products faster, and reducing both application risk and overall operating and engineering costs.

AppSec's Achille's Heel

No finger-pointing or blame game is attached to calling out the elephant in the room, but **95% of vulnerabilities are inadvertently introduced in the development phase.** Despite being completely unintentional, the fact remains that they're there, and they're causing a whole host of problems. In an ideal world, we could eliminate vulnerabilities in the development phase at the source. Eliminating them here would help us create better and more secure software faster, and eradicate most of, if not all, our AppSec vulnerability-related issues in an instant.

Unfortunately, this vulnerability problem isn't new: reducing or eliminating vulnerabilities has always been problematic in software development and AppSec. Developers don't like security training, deem it out of their remit, and are bored by it (most of them, that is). Yet, simple logic would dictate that those who create the problem should own and fix it. Ironically, by the time the software gets to the security team's end of the SDLC, the vulnerabilities now create the bottleneck. Suddenly, it's security's responsibility to fix the bottleneck, and they suffer the inevitable backlash in the holdup. But **security engineers aren't developers, and... we find ourselves in a Catch-22 situation.**

Besides, when security does pass the vulnerabilities back to the dev team to fix, devs are engaged elsewhere on other projects and no longer have the time. The clear answer to this problem has to be to get developers to own their mistakes, remediate the errors as they appear, and prevent them from committing vulnerable code. Do that, and watch your problems evaporate. Unfortunately, this has never happened.

Development and security are like chalk and cheese

At opposite ends of the lifecycle, development and security teams are poles apart in terms of their actual location, approach, skillsets, and mindsets. As **speed** and **security** are trade-offs, every business's challenge is finding the right balance. One main goal of Shift-left was to resolve this very issue: Move security testing closer to development, get them talking and working together, and everything will move much faster and be more secure. Unfortunately, for the most part, Shift-left failed because organizations and their existing monolithic apps were too inflexible. Consequently, though a great idea in principle, this proved a big flaw in the Shift-left idea.

Why security scanning tools could never replace Shift-left

Instead, organizations shifted security scanning tools left. Unfortunately, this went against their fundamental design. Security scanning tools were designed for security testing **near the end of the SDLC**, when the app was almost ready for release, and not for the fast-paced, constantly changing development stage at the start of the lifecycle. Dev teams code fast, need to test frequently, get immediate feedback, and then rinse and repeat. In contrast, security scans are notoriously slow (scans often take several hours and more to run). Given this lack of speed, deploying the tools to work in a fast-paced, constantly-changing environment like development was wishful thinking for two main reasons.

First, as briefly mentioned, developers are not security trained, so the security reports created were of little use (devs don't do pdfs!). Second, expecting devs to run these security tests every time they commit code and then sit there twiddling their thumbs while waiting for the results was a non-starter.

Tools need to be seamless, fast, and accurate

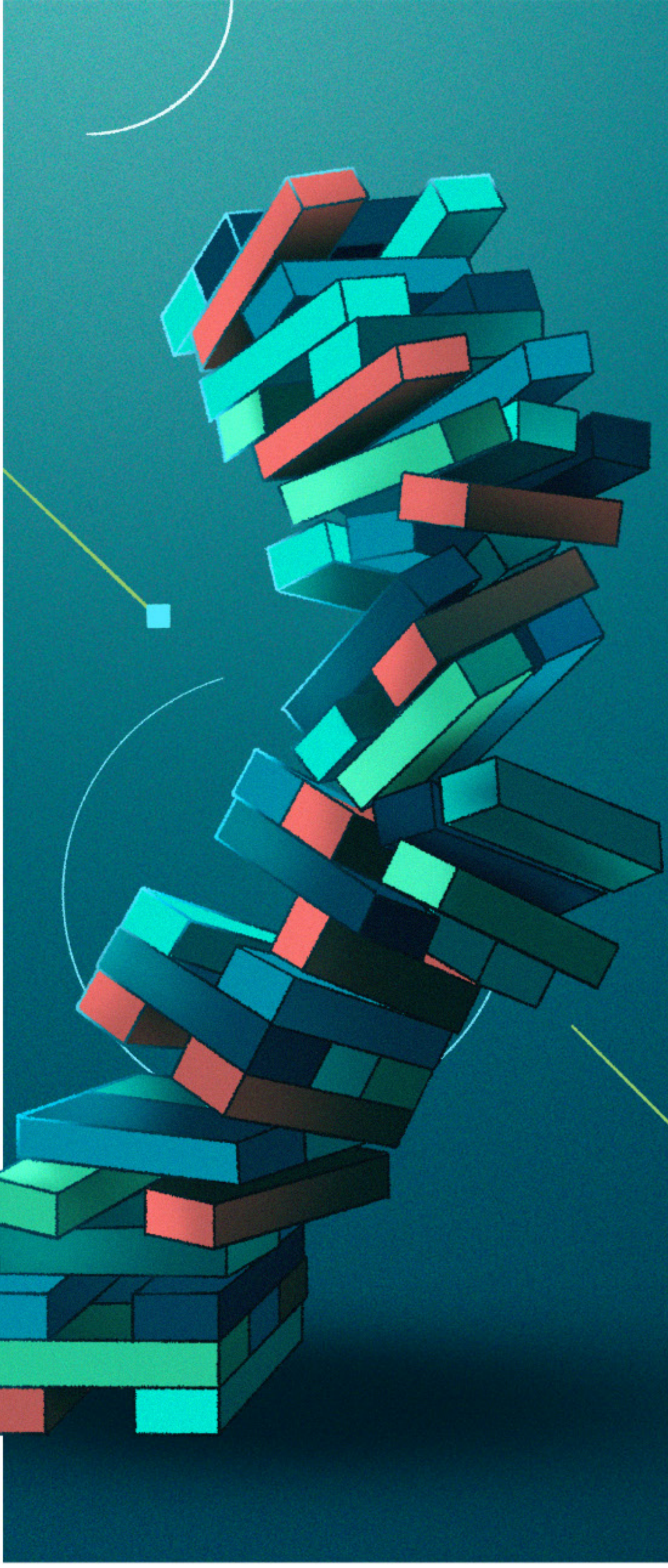
To be adopted, tools need to make the developer's life easier. Tools must work, be fast, accurate, and seamlessly integrate with and complement the developer's workflow. Anything else and the solution won't ever work, and the tools won't ever get used. Regrettably, security scanning tools failed in multiple areas, and **all shifting these tools left did was garner an illusion that security had moved closer to development.** Unfortunately, given no viable alternative, security scanning tools became the de facto Shift-left replacement.

Tools must help, not hinder software development

Being slow was one thing, but security scanning tools overwhelming security teams with noise was another. Yes, comprehensive reports do have their place. But you have to question the validity of serving up a 5,000-page pdf security report that 1) takes hours to create, 2) is riddled with false positives, false negatives (and other noise), 3) is written for security teams and not developers, and 4) is then given to a team that lacks the adequate resources to cope with even a fraction of what the report contains. How a 1, 2, or even 5+ person security team could hope to triage and make sense of a 5,000-page pdf report is anyone's guess.

Point solution tools are too unwieldy

Furthermore, most security scanning tools are point solutions. To cover all bases, organizations would need to deploy multiple tools. That many of these tools proved expensive and complex to set up and maintain goes with the territory. Still, because they rarely integrated with other tools, it added to the security team's burden. (Who among us hasn't experienced the confusing hassle and aggravation of managing multiple devices, dashboards, and programs? Consolidating all into one easy-to-see and manage dashboard would quickly remove the complexity and difficulty and save time.) But for already overwhelmed security teams moving ever closer to alert fatigue, **all point solution tools did was squander more time, increase the risk of human error, and move them closer to burnout.** Moreover, security scanning tools also failed to address organizations' long-standing data protection and AppSec concerns.



Data protection and application security are long-standing concerns

"Almost 80% believe their organization is vulnerable to multi-tiered cyber attacks that can impact the entire application stack in the next 12 months, with 48% noting that an expanded attack surface has posed more challenges." [5]

Again, stats show that 2/3rds of organizations that suffer a data breach do not survive. When you also consider that cybercrime is up by 600%, it's hardly surprising that these are major concerns.

Unfortunately, where data protection and AppSec are concerned, it's like putting the cart before the horse: you need AppSec to protect your data (and need DevSecOps for both going forward). As we'll come onto shortly, it's most definitely possible to buy AppSec, but...

"You can't just buy DevSecOps." [6]

Nonetheless, remediating vulnerabilities is the key to turning your AppSec fortunes around. But the question of why organizations struggle to remediate vulnerabilities remains.

Why do organizations struggle to remediate vulnerabilities?

In addition to what we've already covered, other major problems with remediating vulnerabilities and their effects include:

- **Remediating vulnerabilities is by nature complex and slow**
  Once the vulnerability is committed to the lifecycle, discovering the vulnerability later only adds to its complexity and the number of resources available at the time to resolve it.
- **A lack of experienced resources can lead to an inability to patch software quickly**
  Vulnerable software can lead to disastrous consequences, including system downtime, exposing you to unexpected compliance issues, and increasing operational risk.
- **The cost to fix vulnerabilities increases throughout the lifecycle**
  The later the discovery, the greater the number of resources needed, and the greater the overall cost to fix it (up to 640x more[7]).
- **A difficulty integrating security into the development process**
  Due to various factors, including a lack of available expertise, having siloed teams (and a lack of communication and collaboration between dev and sec), as well as a lack of tools and resources, etc.
- **Tool sprawl and technology overload**
  As already discussed, this can cause issues such as complexity and compatibility issues, time- and resource-wasting, increasing the chance of alert fatigue, weakening your overall security posture by creating unseen security gaps, etc.
- **A lack of development resources**
  When projects are complete, developers usually move on to the next one. Subsequently, discovering a vulnerability much later means the original developers are engaged elsewhere and have zero availability to help. Problems include unexpected delays, impact on other systems and operations, increased vulnerability backlogs, and the risk of non-compliance.
- **A poor developer-to-security engineer ratio**
  Often 100:1 or more, when large lists of vulnerabilities are involved, remediating and clearing the security bottleneck is both difficult and unenviable.
- **A lack of automation**
  Manually monitoring multiple tools and performing security tasks and testing is time-consuming, energy-draining, and always at risk of nudging teams closer to alert fatigue. A lack of automation slows response times, increases the chances of human error, and elevates overall business risk.
- **Developers are not security trained**
  Many do not deem security their responsibility or even within their remit. Trying to train them in security has always been an impossible task.

Other key concerns

Other concerns organizations voiced include protecting against malware and the escalating number of vulnerabilities, threats, and breaches. Sadly, many organizations report that they are ill-equipped to deal with these security concerns highlighting

- A lack of skilled personnel.
- A low-security awareness among both employees and management.
- A lack of security budget.
- Poor inter-departmental communication and collaboration.
- An overall lack of management support.[7]

Shift-left failed and, **by their very design, security scanning tools, could never hope to meet the needs of today's fast-paced software development world.** When you also factor in organizations' ill-preparedness and that no viable security option was ever on the horizon, it's hardly surprising that the future outlook for AppSec has been bleak.

AppSec desperately needed a viable, simple, and proven solution

It was 2017. Having just secured one banking app for a global banking institution, Stefan Streichsbier, GuardRails' CTO, and his team threw a long-awaited party to celebrate. Party hats and balloons galore, it was during the second lap on the conga-around-the-punch-bowl dance when the realization hit that it'd taken almost 2-years to secure that one single app.

Admittedly, that particular project's deployment speed was normal for the time but, as celebratory as that event was, unfortunately, continuing with that same trajectory, it'd be around the year 2083 before they secured the bank's 31+ other untouched and insecure apps (give or take the odd leap year or 2).

"Yes, you've secured the crown jewels, but how will you secure the rest?

From there, I looked at the existing tools, but none were fit for purpose. Then, I decided to take my own approach and make something that resonates and works for developers... "

Stefan Streichsbier | GuardRails CTO and co-founder

Fortunately, that was a clear indicator that AppSec needed a new and much simpler solution. Today, industry changes and global events have reinforced that very fact. For any AppSec solution to be universally accepted and adopted, it had to be automated, fast, easy to use, seamless with, and complement, the developer's workflow. Above all, the solution had to be simple and effective.

The GuardRails 3-step simple AppSec solution

DevSecOps is primarily about people, processes, and technology. But effective DevSecOps is automated, and that means selecting the right tools to support how you do business. GuardRails is such a tool. To deliver what Shift-left couldn't, the solution needed to be simple.

Our solution comprises 3-steps:

1 **Scan commits for code changes and vulnerabilities** – This immediately identifies any vulnerabilities in the code and prevents the developer from committing the source code.

2 **Immediately flag the vulnerability and notify the developer** – The vulnerability is automatically assigned to the developer to fix, and they are notified immediately.

3 **Simultaneously provide detailed, context-specific remediation advice in the notification to aid the developer in fixing the vulnerability** – Of course, if the developer knows how to remediate the vulnerability, they can safely ignore the advice and immediately fix it. But, if they don't know how to fix it, a direct link to the help material aids them in self-educating and fixing the problem quickly.

Ultimately, the GuardRails system ensures that developers cannot successfully commit vulnerable code. Because this occurs seamlessly within the developer's workflow, they can immediately fix the vulnerability with minimal effort and without losing focus.

In its barest form, that's how GuardRails does it.

(Too simple? We know you'll have questions about the above, so we've included an FAQ section on page 10.) However, one burning question we know you'll have is:

How effective is the GuardRails solution?

Though we could wax lyrical about how good the GuardRails solution is, that one's best coming from others:

raya

"Within 6 months of using GuardRails, our pen test findings have been reduced by 50%."
Stepanus Mangunsong, Bank Raya

AirAsia

"Now with GuardRails, the bottleneck has been reduced, making our management very pleased, allowing us to go to market faster, allowing my security testers to focus more on the high-level bugs and vulnerabilities now that the baseline is enforced."
Fabrice Marie, AirAsia

V-KEY

"With GuardRails in place, I can sleep better now because I know that GuardRails is there to help me ensure compliance with our DevSecOps process."
Chiang Kai, VKey

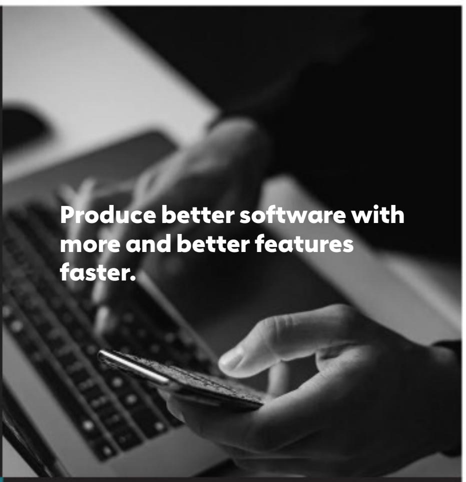
How does GuardRails benefit your organization?

GuardRails was built to help developers meet the needs of today's AppSec. By integrating security testing into all aspects of development, GuardRails delivers what Shift-left failed to do. Remediating new vulnerabilities at source prevents them from proceeding into the rest of the SDLC. (You'll note we say new vulnerabilities. If you're wondering how GuardRails manages your existing vulnerabilities, we cover this in the FAQs on page 10.)

Remediating vulnerabilities at source delivers significant benefits, including allowing you to:

Create more secure software faster and accelerate your time to market.

Eliminate your security bottleneck and alleviate the pressure on your security team (likely for the first time), allowing them to focus on more pressing security issues.



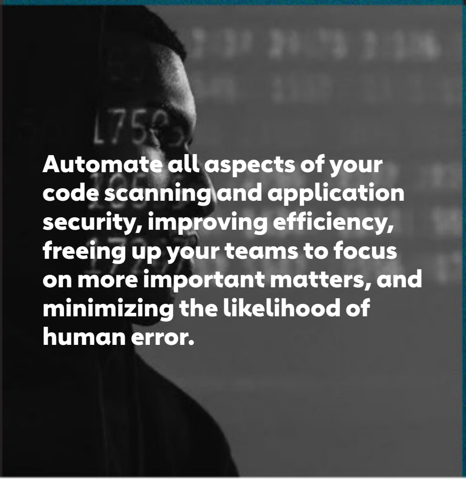
Produce better software with more and better features faster.

Reduce engineering costs through real-time feedback as critical issues are introduced, as well as advising software engineers on how to fix them.

Slash production costs and bottlenecks through fewer holdups and slicker pipelines.

Improve your flexibility and critical response times and dramatically reduce the cost of bug fixes.

Upskill and educate your developers to improve overall development efficiency, increase morale, and promote better and more solid results.



Automate all aspects of your code scanning and application security, improving efficiency, freeing up your teams to focus on more important matters, and minimizing the likelihood of human error.

Automatically protect all current and future repositories. This not only leads to less work for your security teams but also provides that essential peace of mind and breathing space to focus on implementing the rest of DevSecOps, and lowers your Total Cost of Ownership (TCO).

What makes GuardRails different?

GuardRails delivers Shift-left right out of the box. In addition to our 3-step simple process, here's how we do it, what we provide, and some insights into our approach:

- **Real-time Source Code Analysis** – Improves your developer's ability to remediate vulnerabilities as they occur. This correction also helps to mitigate the risk of major vulnerabilities being pushed further into development and causing more complex and costly problems later on, and reducing or even eliminating your security bottleneck.

- **Just-in-Time (JIT) Training** – Provides real-time guidance and direction on fixing issues and vulnerabilities. Tight integration within the developer's workflow eliminates distractions and, by including direct links to topic-specific training, instantly removes the developer's long-standing dislike of security. JIT training also educates and upskills your team to deliver bug-free code much faster.

- **True Left Vision** – GuardRails delivers what Shift-left failed to achieve: True Left. Moving security testing left and as close to development as possible by embedding security testing throughout your workflows.

- **Developer Impact and Responsibility** – GuardRails provides developers with the tools to not only take responsibility for and help implement a culture of security within the organization but also to take ownership of their mistakes, upskill and self-educate, and help build better and more secure software faster.

- **Unified Security Platform (a single pane of glass)** – As discussed, existing technology is fragmented, isolated, and difficult to configure and manage (and often requires dedicated teams.) Moreover, it's time-consuming, resource-intensive, and increasingly complex. Combining all components into a fast, well-tuned, and configurable single pane of glass security dashboard that automates security tooling, scanning, and CI pipeline management makes the entire approach much simpler, easier, and more effective.

- **Opinionated Approach to Security** – By taking a much more opinionated, simple, logical, and practical approach, we filter out the noise and report only the essential issues. By alleviating overwhelm and removing the "Cry wolf" syndrome that has distracted security teams worldwide for years, you finally get the calm breathing space you need.

- **Version Control System Integration** – GuardRails monitors your entire development environment, constantly scanning for changes, comparing files, etc. When a change is detected, GuardRails automatically takes all appropriate actions to maintain your development velocity, keep your devs in their workflow, and protect your end-to-end security.

- **Security Tool Orchestration** – GuardRails orchestrates and automates over 40 security tools to present the required information in one customizable, easy-to-use dashboard that takes the pressure off of everyone.

- **Security Rules Curation** – Smart rules curation examines every single rule of every single tool to determine whether it qualifies as a security issue. It's fast, and when it detects a security issue, it flags it automatically and immediately. That way, in-workflow reports contain only relevant content, are succinct, and your teams only get notified on important matters.

- **False Positive Detection** – GuardRails' proprietary false positive detection system has an exceptional accuracy rate, helps eliminate irrelevant security issues and false alerts, and provides a feedback loop to keep our detection system current, accurate, and safe.

- **Security Testing Techniques** – GuardRails orchestrates and unifies over 40 tools that perform Static Application Security Testing (SAST), Software Composition Analysis (SCA), Secrets Detection, and Infrastructure as Code (IaC) to keep you secure from code-to-cloud.

Frequently Asked Questions

Below are the most common questions. If you have a question that isn't covered, please contact us; we're always happy to help.

- **Many security tools are slow, so how fast is GuardRails?** – GuardRails is very fast. We have built smart scanning capabilities that scan for new code changes (rather than the entire code base) and only flag/notify new errors. As such, most scans take only seconds.

- **How secure is GuardRails?** – GuardRails uses GuardRails to secure GuardRails. We are also ISO 27001 and SOC Type 2 certified.

- **How accurate are you at finding vulnerabilities?** – Very accurate. When a developer commits code, GuardRails immediately scans for code changes. On detection, we flag the vulnerability and notify the developer in real time and within their workflow. We then record this information in the Version Control System (VCS), where it can be accessed and reused for review/training purposes, etc.

- **How easy is GuardRails to set up and use?** – GuardRails installs in seconds and works out-of-the-box with no additional configuration required. (Though GuardRails is fully customisable.)

- **What do you mean by a single-pane-of-glass?** – This is our single consolidated dashboard that combines all components from our security engines and languages into one place. This includes threat overview, opened versus fixed vulnerabilities, most common vulnerabilities, vulnerabilities by language, etc.

- **Is the onboarding/training complex and time-consuming?** – No, it's highly intuitive and educational by design. Because we provide in-workflow, real-time, context-specific help, we overcome and eliminate the need for developers to sit through hour-long+ (boring, to them) mandatory security training videos. Moreover, because we have deep integration with Bitbucket/GitLab/GitHub, this allows us to wrap security around what is already familiar VCS functionality.

- **Can you elaborate on your context and environmentally specific education and training?** – By providing context, language, and environmentally-specific help, we empower developers to [finally] take ownership of and fix their own errors. Feedback and results show this completely overcomes the long-standing developer resistance to security training, helps them become better developers faster, and solves that 'forever' problem mentioned earlier. (The results clearly show that it's not just developers who love GuardRails; security engineers do, too!)

- **How difficult/complex is GuardRails to set up and maintain?** – Neither. It's easy and simple and takes only seconds. Once installed, GuardRails rapidly scans your repositories, immediately flags any existing vulnerabilities, and then we're ready to go.

- **How scalable is GuardRails?** – Very. GuardRails integrates with all modern VCS, automatically scans all new repositories, and scales automatically and in parallel with you.

- **What languages are supported?** – Supported languages include Apex, Dotnet, Python, Ruby, PHP, JavaScript, TypeScript, Rust, Kubernetes, Go, Solidity, Java, Elixir, Terraform, and C/C++.

- **How customizable is GuardRails?** – Because GuardRails takes an opinionated approach to security, it works for most use cases out-of-the-box after installation. Almost every setting in GuardRails can be completely customized, whether adding custom engine rules or tweaking the platform settings. Indeed, if you have specific needs, such as we see with in-house developed apps, you can even create your own custom scanning engines.We provide full documentation but are always happy to help.

- **We know we have many existing vulnerabilities; can you explain how GuardRails deals with them?** – We use the *Stop the Bleeding* approach (you may remember this approach as one of First Aid's priorities: to prevent the patient's condition from worsening). As our CEO, Andrew Duck, succinctly put it,

"Once installed, we may detect 1,000 existing vulnerabilities, but GuardRails will ensure you never get to 1,001."

Next steps

If what you've read in this paper interests you, the next step would be to see GuardRails in action. We have a short, live, power-packed demo that only takes about 15 minutes. First, you'll see how quick and easy installing GuardRails in real-time is. Second, you'll see how easy and intuitive GuardRails is to use. Third, you'll see why GuardRails is not only turning the AppSec industry on its head but also finally *Putting the Sec in DevSecOps*.

Book a Demo

Bibliography and References

[1] Fortinet, "IDC FutureScape 2023," Fortinet, 2023. [Online]. Available: https://www.idc.com/events/futurescape. [Accessed 25 January 2023].

[2] Bischoff, P, "How data breaches affect stock market share prices" Comparitech, 9 February 2021. [Online] Available: "https://www.comparitech.com/blog/information-security/data-breach-share-price-analysis/. [Accessed 13 February 2023]

[3] R. Tarun, "Top Cybersecurity Challenges for CISOs to Address in 2023," Fortinet, 15 December 2022. [Online]. Available: https://www.fortinet.com/blog/ciso-collective/top-cybersecurity-challenges-for-cisos-to-address-in-2023. [Accessed 25 January 2023].

[4] C. Osborne, "The more cybersecurity tools an enterprise deploys, the less effective their defense is," ZDNet, 30 June 2020. [Online]. Available: https://www.zdnet.com/article/the-more-cybersecurity-tools-an-enterprise-deploys-the-less-effective-their-defense-is/. [Accessed 20 January 2023].

[5] E. Yu, "Firms fear software stack breach as attack surface widens," ZDNet, 1 February 2023. [Online]. Available: https://www.zdnet.com/article/firms-fear-software-stack-breach-as-attack-surface-widens/. [Accessed 2 February 2023].

[6] W. Kelly, "DevSecOps: 5 tips for seeding a culture transformation," RedHat, 18 August 2022. [Online]. Available: https://www.redhat.com/architect/devsecops-culture. [Accessed 24 August 2022].

[7] C. Jones, Applied Software Measurement: Global Analysis of Productivity and Quality, 3 ed., New York: McGraw Hill, 2008, p. 697.

[8] Tripwire, "Application Security Report 2022: Key Trends and Challenges," Tripwire, 8 August 2022. [Online]. Available: https://www.tripwire.com/state-of-security/application-security-report-2022-key-trends-and-challenges. [Accessed 20 January 2023].

[9] PurpleSec, "Cyber Security Statistics The Ultimate List Of Stats Data, & Trends For 2022," PurpleSec, 17 October 2022. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/#Cybercrime. [Accessed 20 January 2023].

[10] Privitar, "New Privitar Survey Reveals Business Opportunity to Build Consumer Loyalty Through Data Privacy," Privitar, 26 August 2020. [Online]. Available: https://www.privitar.com/press-releases/new-privitar-survey-reveals-business-opportunity-to-build-consumer-loyalty-through-data-privacy/. [Accessed 27 July 2021].

[11] C. Griffiths, "The Latest 2023 Cyber Crime Statistics," Aag, 6 January 2023. [Online]. Available: https://aag-it.com/the-latest-cyber-crime-statistics/. [Accessed 20 January 2023].

[12] M. Hales, "8 big DevSecOps challenges and how to overcome them," Adapatavist, 21 December 2021. [Online]. Available: https://www.adaptavist.com/blog/8-common-devsecops-challenges-and-how-to-overcome-them. [Accessed 25 January 2023].

[13] M. Nawale, "The Top Challenges Faced by Organizations Implementing DevSecOps," ZScaler, 18 May 2022. [Online]. Available: https://www.zscaler.com/blogs/product-insights/top-challenges-faced-organizations-implementing-devsecops. [Accessed 23 January 2023].

GUARDRAILS

GuardRails is an end-to-end security platform that empowers developers to find, fix, and prevent vulnerabilities in their web and mobile applications.

Trusted by hundreds of teams around the world to build safer apps, we easily integrate into the developers' workflow, quietly scan as they code, and show how to fix security issues on the spot via Just-in-Time training. We keep the noise low and only report high-impact vulnerabilities that are relevant to your organization.

GuardRails helps you shift security everywhere and build a strong DevSecOps pipeline, so you can go faster to market without risking security.

www.guardrails.io