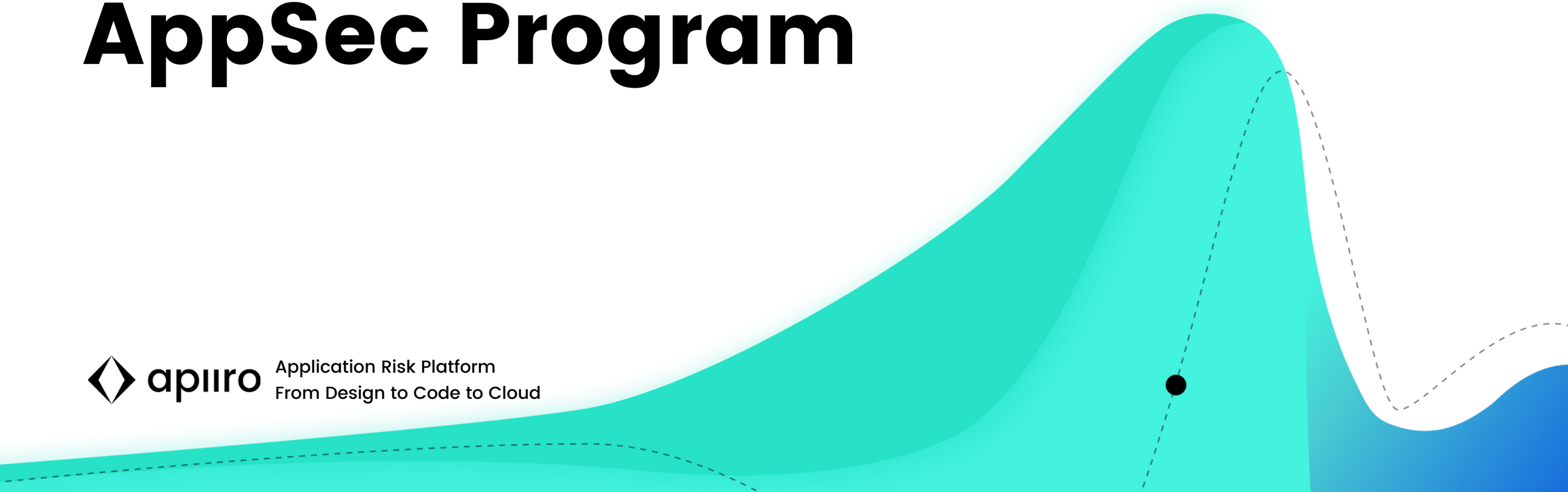
6 Steps to

Build & Scale a Risk-Based AppSec Program

RSAC Innovation Sandbox 2021 WINNER

Gartner COOL VENDOR 2021



apiiro

Application Risk Platform
From Design to Code to Cloud

Welcome!

Reducing your application security and privacy risk is a never-ending process. It involves multiple teams, from business to development to security to compliance, working together to understand and effectively manage risk. Unfortunately, CISOs often find themselves speaking a different language than their CIO and VP of Engineering, leading to a lack of innovation and progress. But with a common framework, it's possible to remedy those differences and achieve laser-focused alignment across teams.

There are many strong Application Security Models, such as the Building Security In Maturity Model (BSIMM) and the OWASP Software Assurance Maturity Model (SAMM). These go into extreme technical detail and we aren't looking to supplant them. At the same time, we believe that existing Application Security models fall short by focusing only on code vulnerabilities without the full context needed to understand risk.

A deep understanding of the code is essential, but risk can never be understood in isolation. It is multidimensional and requires a broad perspective and analysis of everything from the user story to the individual developer expertise to the application components to the configuration of your cloud infrastructure. In other words, to truly understand and effectively manage your risk, you need to connect the dots across the entire SDLC, from Design to Code to Cloud.

Building an AppSec program with the legacy tools you're comfortable with may feel safe but is fundamentally misguided. Plugging in SAST and SCA tools without context and being overwhelmed with alerts will only result in wasted time and frustration across your organization.

We established a simple framework to create and manage a measurable, Risk-Based Application Security Program to help organizations improve their program at all stages - for those who are just getting started and for those looking to take advantage of the latest best practices and technologies. There are no requirements other than an inquisitive nature and a willingness to challenge current thinking. If you're ready to learn how to gain an understanding of multidimensional application risk: read on.

Idan Plotnik,
Co-Founder & CEO

apiiro

Table of Contents

6 Steps to Build & Scale a Risk-Based AppSec Program

Overview

An Application Security or Application Risk Program is not a set of technologies. It is a collection of people, processes, and technologies that are seamlessly intertwined and work together in order to reduce risk, lower costs, and deliver faster. For many years, AppSec programs have focused on vulnerabilities, from SQL Injection to Cross-Site Scripting (XSS), but a modern understanding of application and infrastructure security is risk-based and focused on business impact.

Ask 50 CISOs or Application Security Engineers what an AppSec program should look like and you'll get 50 different answers. Every organization has unique needs to define how Security is integrated into their Software Development Lifecycle (SDLC), often called the Secure SDLC (SSDLC) or the Secure Development Lifecycle (SDL).

AppSec Practitioners are Overworked & Overwhelmed

Application Security Architects are outnumbered by Developers by a median ratio of 159 to 1, with each developer committing multiple changes a day. To release these changes to production, development teams need to go through security and compliance reviews, pen-tests and risk assessments, which are manual, periodic, and inaccurate because they're based on self-attestation.

A single architect is responsible for reviewing hundreds of changes a day and deciding which are risky - an impossible task. On top of that, the same person is responsible for triaging noisy alerts from SAST, SCA, and other application and cloud security tools

Code Risk is Multidimensional

A multidimensional Application Security & Risk Program must also extend from Design through Production. Prioritizing risks correctly requires going beyond code because risk is multi-dimensional.

Consider a developer that changes the logic of a sensitive internet-facing API and adds PII. The security expertise of the developer matters! The same is true for ticketing systems data, Pull Request discussions, cloud firewall changes, & authorization controls in the cloud API Gateway. These are all critical pieces of information needed to evaluate the risk of a change.

The Path Forward

This guide will help you up-level your program from being focused on Application Security to deeply understanding and acting on Application Risk at a business level. By following this approach, you will accelerate your application delivery while reducing both cost and risk.

6 Steps to Build & Scale a Risk-Based AppSec Program

While building a mature and measurable Application Security Program may seem like an extensive and long-term undertaking, it's not as daunting as you may think. If you follow a **series of six simple steps** you will be able to build a mature, results-based program.

Each of these steps builds on the previous to help bring better understanding and further focus to ultimately help you make better decisions - more efficiently and even automatically.

Even if you have a functioning AppSec program, don't skip ahead! Take a fresh look at your goals by starting from the beginning and you may find some areas to improve incrementally and others to rebuild from scratch.

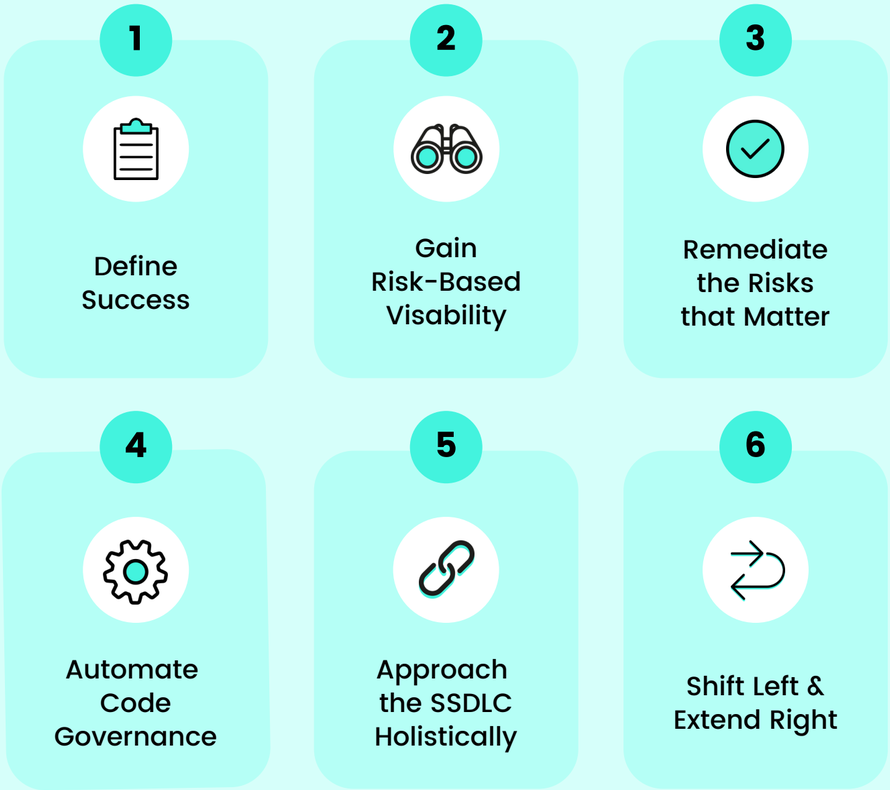


Illustration 1

6 Steps to Build a Risk-Based Application Security Program

01 Define Success

The first step in creating an Application Security (or just about any other type of program) is to define success. Too often, key metrics include items such as:

- Number of open vulnerabilities
- Number of vulnerabilities remediated in the past 30 days
- Mean time-to-resolution

But these are vanity metrics. Remediating the wrong vulnerability quickly is not a measure of success. These types of metrics allow people to have the illusion of showing progress or security but do not tie to any measurement your executes or Board of Directors care about.

"Defining success" means thinking at a strategic level about what your Application Security program is trying to accomplish and how it fits in with your other goals, such as driving revenue and retaining talent. A successful Application Security program cannot be focused on tactical measurements of vulnerabilities and weaknesses. It needs to see the whole picture: from application code to Infrastructure-as-Code, to the settings on your API Gateways. It is impossible to understand and intelligently act on risk in isolation when risk is multidimensional.

The key to defining success well is to focus on the outcomes that will impact your business. Think about the **business impact** of a breach. If data is exposed or stolen, what type of data is it, and how would an exposure impact your customers and your business?

RECOMMENDATIONS

Create a set of metrics that are risk-based, measurable, and combine vulnerabilities and security weaknesses with business impact.

THE BOTTOM LINE

Measurement drives behavior. When you define your metrics to be risk-driven, you set your Application Security program up for success from the start.

02 Gain Risk-Based Visibility

Application and Infrastructure code

Better information leads to better decision-making. That's not a particularly bold statement. But at the same time, we have a tendency to look for data in our narrow area and then... just more of it. More fields. More reports. More dashboards. We don't often take the opportunity to step back and re-evaluate what you'd actually need to make smarter decisions. Asking "How can I gain complete risk visibility into my applications and infrastructure?" is not a simple question and it deserves more than a simple answer.

True application risk visibility requires end-to-end clarity, from design to code to cloud. Data gathering throughout all stages can help make better decisions early in the development process, improving security & reducing unnecessary rework.

Visibility into Application Code and Infrastructure

Technology sprawl has a long-term strategic impact. Development, compliance issues across frameworks and languages. Using more than one technology for key management, authentication, authorization, and encryption leads to unnecessary fragmentation in adjacent areas as well. Visibility into all of your code components is absolutely essential and required to understand your risk, but there's more to the equation.

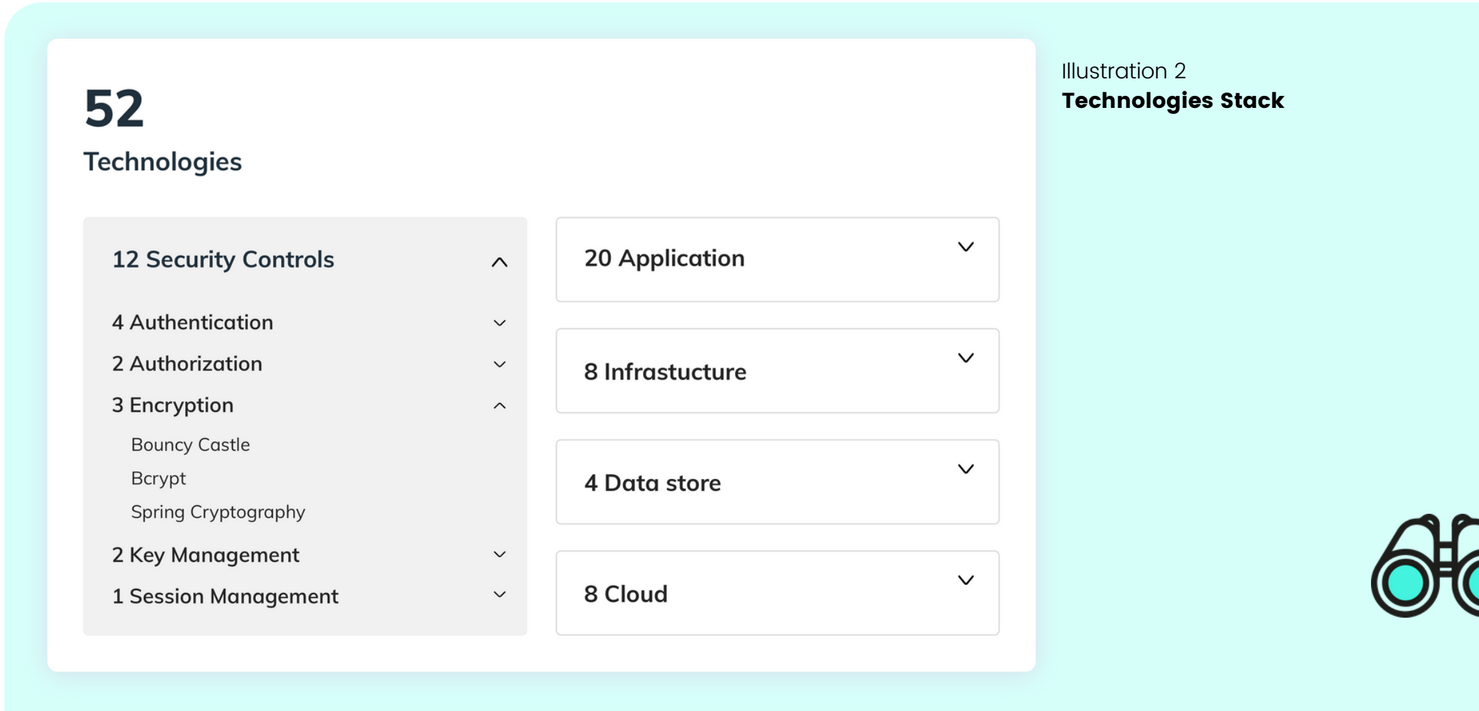


Illustration 2
Technologies Stack

02

Gain Risk-Based Visibility

Contributors and security champions

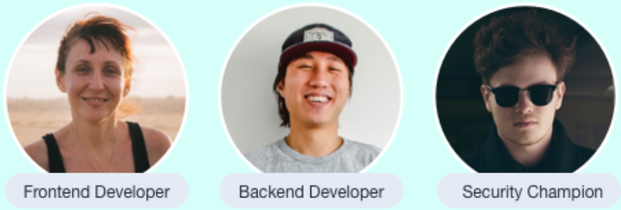
Visibility into Contributors

Understanding the code itself is essential, but unfortunately, that's usually where the thinking stops. To have a true business-level understanding of application and cloud risk, we need to expand beyond technology and code. Building knowledge and activity profiles for each developer based on historical code changes can help with better decision-making.

This includes how many code changes they have committed, whether those were security-related changes, and if they had a business impact. We can also take into consideration data handling, deployment location, and internet exposure, among many other factors. Also consider how greater visibility into your developers' expertise can help improve both the security and efficiency of your development process.

When a risky material change is discovered, it's essential to find the best person to investigate and remediate the issue. You need to understand which developers have experience in specific technologies, languages, security controls, and even specific types of features. On top of that you need to understand the roles of other contributors, including QA, DevOps, Product Management, etc. and how they contribute to the application. In other words, to find the right person for the job, you need to know the person as well as the job! "Asking around" is not an acceptable - or scalable - solution.

Illustration 3
Roles



02

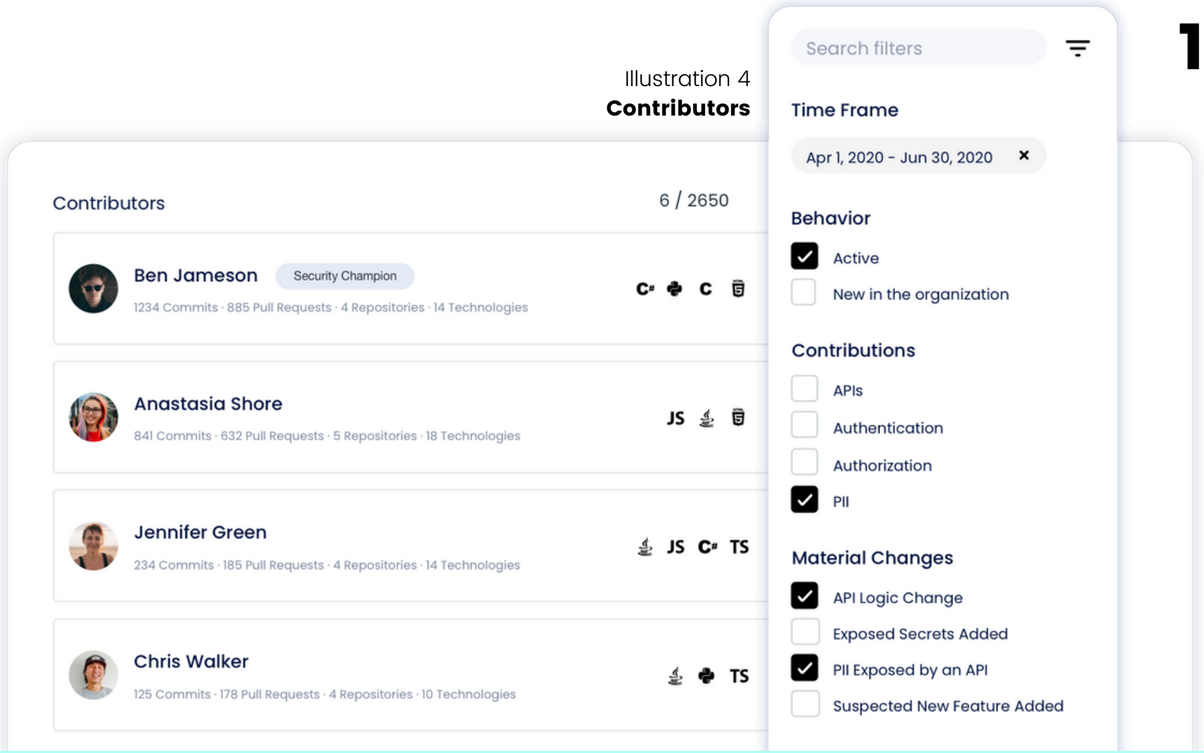
Gain Risk-Based Visibility

Contributors and security champions

Identifying Security Champions

In addition, many organizations are building a Security Champions Program, where knowledgeable security experts are embedded into each development team. A deep-understanding of your developers' strengths and weaknesses is critical to the success of the program. Your Security Champions should have a skill set that fits with their teams so they can provide the right coaching to the right people. They can also better understand when their expertise is required.

Illustration 4
Contributors



RECOMMENDATIONS

Create an inventory of your Application code, infra-as-code, contributors, and more. Ensure that it is updated in real-time and leverages automation to ensure accuracy.

THE BOTTOM LINE

Don't limit yourself to what's in front of you. There is tremendous potential to improve your visibility into not only your code but the **context** that surrounds it - and with it, your understanding of application & business risk.

03 Remediate the Risks that Matter

When thinking about application risk, we have a tendency to focus on - and limit ourselves to - vulnerabilities and detection tools like SAST, SCA, DAST, IAST or even orchestration tools that connect them together. Relying on these tools is fundamentally misguided because they lead to significant noise and false positives. Scan "findings" are assigned to people for remediation without any regard to the context that affects the real-world risk of the finding.

For example, a SQL Injection finding may have a "Critical" scan score but that label doesn't tell the whole story. Is the application Internet-facing? Does the application contain PII? If so, this PII is exposed by an Internet-facing API with proper security controls? In many cases, a "vulnerability" with a score of 5 can be much more significant than another with a score of 10, depending on the context.

> "Stop trying to remove all unknown vulnerabilities in custom code, which increases false positives. Instead, focus developers on those with the highest severity and confidence."*

Gartner. "12 Things to Get Right for Successful DevSecOps" By Analysts Neil MacDonald, Dale Gardner. 19 Dec 2019

Single Dimensional Analysis

During the SDLC, companies trigger the execution of tools at the specific stages where those tools fit (e.g., SAST at the CI/CD). SAST scans the code at a single point in time, building all possible permutations of code flows, but focusing only on vulnerabilities and disregarding the code components, data, security controls, deployment location, developer experience, and business impact. This results in the discovery of massive numbers of "issues" - a good percentage of which will be false positives, leading to wasted time and frustration for both Security and Development teams.

Another difficulty can arise by using Software Composition Analysis (SCA) tools that are used to discover, govern, and monitor OSS licenses, and security vulnerabilities in dependencies.

These tools also lack a multi-dimensional approach and look only at the included package. They don't know if the code is using the OSS vulnerability code path. In addition, they don't take into consideration if it's a sensitive dependency because of its essence (e.g., an authentication framework) or the experience of the developer that added it to the source code.

03 Remediate the Risks that Matter

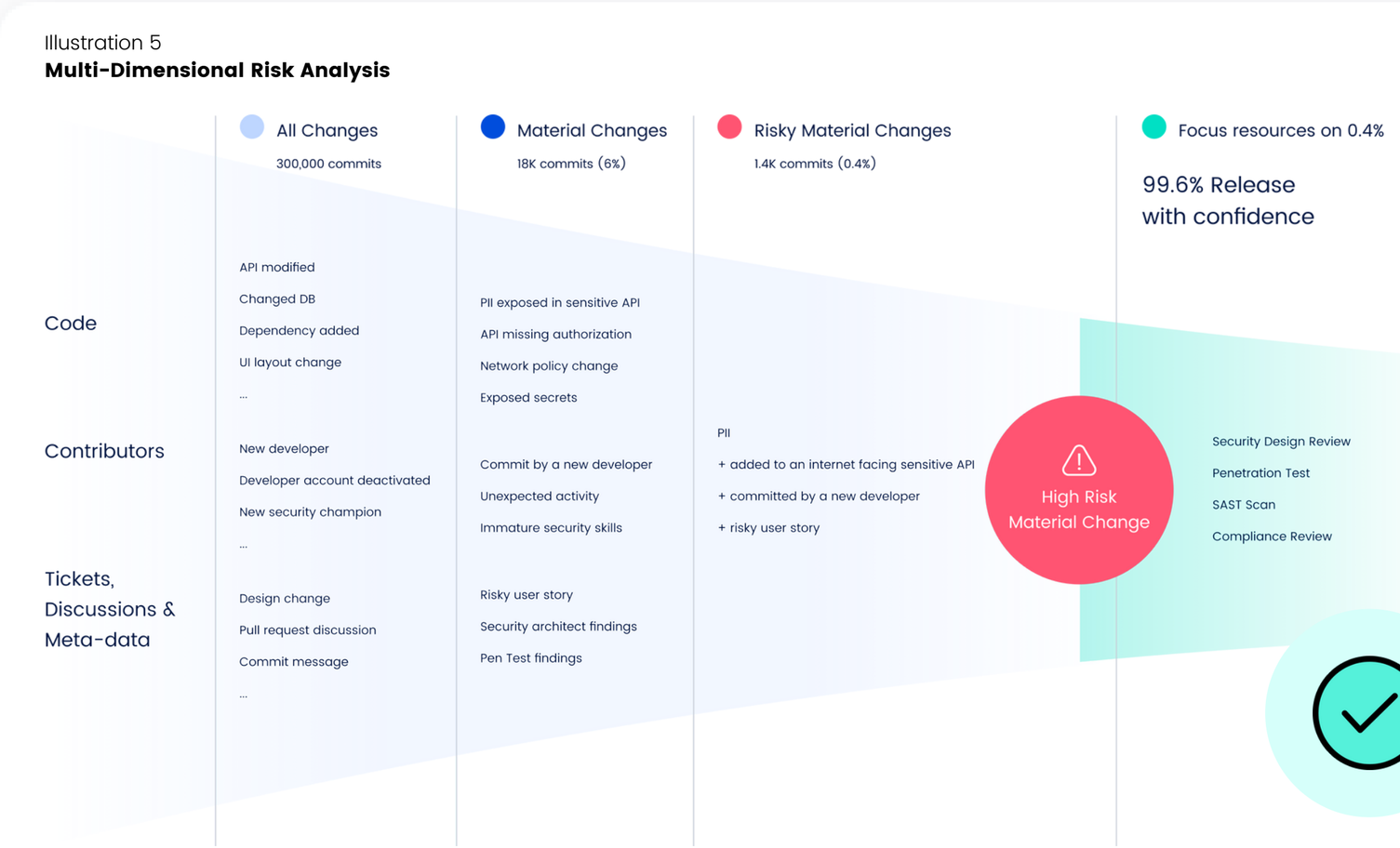
Multi-Dimensional Risk Analysis
Instead of viewing risk in terms of vulnerabilities, focus on the actual risks to your business. This requires an extensive understanding of context. We can classify different types of Risk Dimensions.

A multi-dimensional approach to Code Risk Analysis can reduce and optimize security processes by focusing SDLC tools and processes on what we call "Risky Material Changes." Combining the above factors and more, you can produce a Multi-dimensional Risk Analysis using a contextual model that will help Security Architects and developers focus on the changes that matter most.

RECOMMENDATIONS
Build a unified risk remediation work-plan that encompasses all relevant contextual factors.

THE BOTTOM LINE
To truly understand your risk, prioritize and remediate the risks that matter, you need to take a multidimensional approach.

Illustration 5
Multi-Dimensional Risk Analysis



04 Automate Code Governance

In order to be scalable, an Application Security program needs to be intelligently automated. With a multitude of changes occurring across the SDLC, Application Security Architects and Engineers cannot manually investigate every update and start every SSDLC process.

But there are code governance rules that can be automated to make sure that the right processes are triggered at the appropriate time and with the information needed for those processes to be successful. Consider a change where PII is added to a data model that is exposed by an API that is not protected by an API Gateway. A change of this type should not rely on manual discovery but should automatically:

- **Trigger** Compliance Review or Security Code Review
  Identify & assign the relevant developers and resources, such as a Security Champion
- **Identify** and assign the relevant developers and resources, such as a Security Champion
- **Provide** the contextual information for the developers to fully understand the risk

Consistency Requires Automation
Developers need to understand your organization's risk tolerance and which types of changes will trigger security reviews. But Security Architects are human, and whenever people make decisions, there is inconsistency. Automating code governance with a clearly-defined set of rules removes ambiguity. Development teams can learn from these rules to avoid unnecessary reviews. This is true Shift Left.

RECOMMENDATIONS
Define and deploy workflows that automatically trigger the appropriate SSDLC processes with the context needed to optimize each task.

THE BOTTOM LINE
Automating your code governance will save time and ensure consistency, making it a win/win for Security & Development teams.

05 Approach the SSDLC Holistically

Today, everything is code! From developing the application logic, adding PII to a Data Model, changing a network policy, adding IAM roles, to publishing a new API in the cloud API gateway and configuring authorization control.

There are multiple Secure Software Development Lifecycle (SSDLC) process gates, each functioning independently of the others:

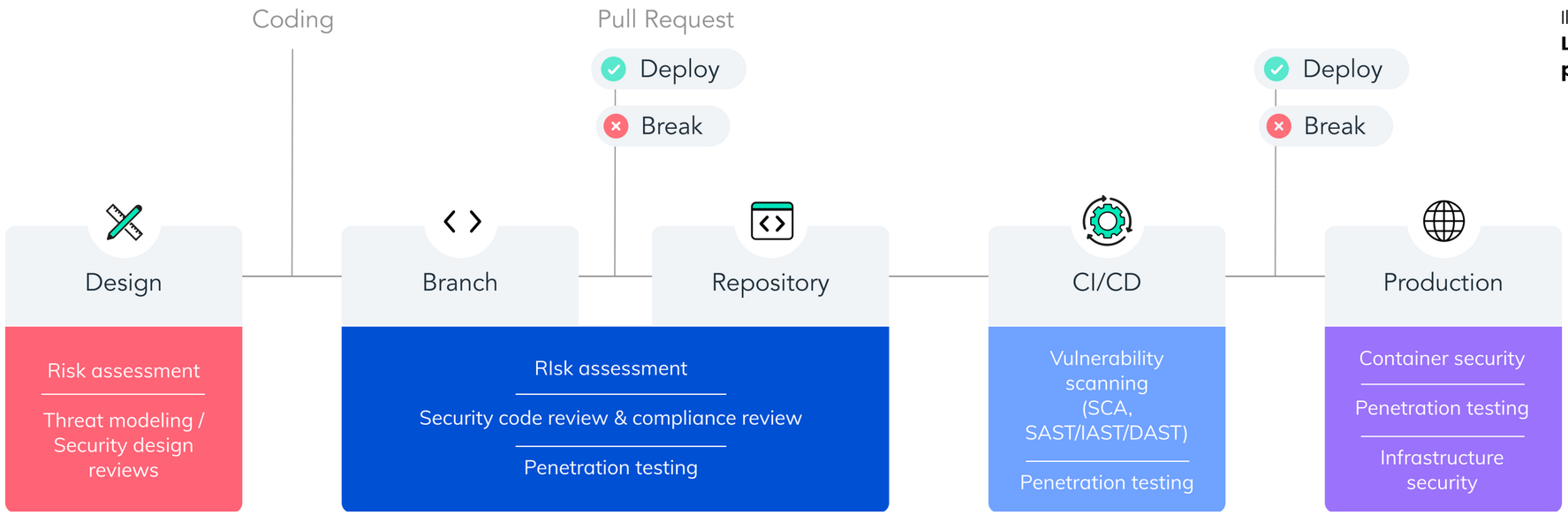


Illustration 6
Looking at the SSDLC across processes and tools

05 Approach the SSDLC Holistically

"Integrating security into DevOps to deliver DevSecOps demands changed mindsets, processes and technologies."*

Most of the time, we examine each gate without looking at the full context across all development stages. Material code changes and configurations need to combine in order to create a vulnerability. This requires taking into account many factors, from design to code to production (e.g., the schema of the data type to the validation of inputs to the security controls on the production systems).

Without an end-to-end view, developers and security architects waste time triaging results and investigating false positives, which leads to lack of trust in those tools.

A Single Pane of Glass Across Processes and Tools
In order to effectively manage a risk-based SSDLC, you need a single view into your entire development process. When you have tools for SAST, DAST, IAST, you have to manage alerts from multiple places, regularly context switching and not seeing the full picture of your risk. You need to not only understand context across all of these tools, but you need to have a single view that correlates & prioritizes vulnerabilities and weaknesses and risk across all tools and processes.

RECOMMENDATIONS
Deploy a solution that analyzes, prioritizes and helps you manage risk across the entire SSDLC in a single pane of glass.

THE BOTTOM LINE
Looking at the SSDLC across processes and tools from a single pane of glass is required to make better, more contextual decisions.

Source: Gartner. "12 Things to Get Right for Successful DevSecOps" By Analysts Neil MacDonald, Dale Gardner. 19 December 2019

06

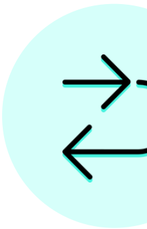
Shift Left & Extend Right

While “Shift Left” has been a buzzword for some time in the developer and security communities, our execution as an industry hasn’t met expectations. Identifying vulnerabilities earlier in the SDLC is an improvement, but a true risk-based, Shift Left approach will focus on risky material changes before they even become vulnerabilities. It also recognizes that the information across the entire SDLC is relevant to understanding risk in any individual stage, so we need to extend our data gathering “to the right” to include Infrastructure as Code.

True Shift Left Starts with Developer Training

Continuous feedback is the key to up-leveling the security knowledge of developers by orders of magnitude. In an ideal world, all developers would be trained and experienced in secure coding practices from front-end to back-end and be skilled in preventing everything from SQL injection to authorization framework exploits. Developers would also have all the information they need to make security-related decisions early in the design phase. Once again, reality falls short of the ideal. While CI/CD automation has given developers ownership over the deployment of their code, those developers are still hampered by a lack of visibility into relevant information that would help them make better decisions before even sitting in front of a keyboard to write code. The best Shift Left approach is to provide developers with the information and training they need to prevent potential risks from becoming vulnerabilities in the first place.

Security training is another example of how organizations are applying a “check the box” mentality - instead of collaborative and contextual mentality towards security and compliance. When vulnerabilities are found in code, there is often after-the-fact training to ensure that developers learn how to avoid the same error in the future. A better approach would be to understand the developer’s experience, skill set, and what they are trying to accomplish in the context of the application, from technologies and frameworks used to APIs.



06 Shift Left & Extend Right

If a developer is working on a new type of security control they haven't worked on before, an organization should provide the appropriate training **before** there is a security issue. The same thing applies to new languages, technologies, and areas of the stack. Going even further, a contextual understanding of everything from the Jira ticket to production settings would allow organizations to provide the right training at the right time to the right people. THAT is true Shift Left security.

Extend Right

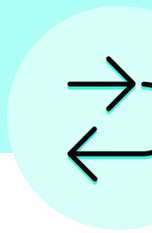
While "Shift Left" has received most of the attention, a comprehensive approach to application risk requires you to also "Extend Right". Consider a developer that is assigned to add PII fields to an Internet-facing API. The authorization controls in the Cloud API Gateway are critical to the security of the new feature! "Shifting Left and Extending right" doesn't mean that a scanning tool or Security Architect should detect a security risk earlier in the process - it means that a developer should have all the context to prevent the vulnerability before it even occurs.

RECOMMENDATIONS

Build developer training and Security Champion programs that are contextual and developer-first.

THE BOTTOM LINE

True Shift Left & Extend Right security goes beyond identifying vulnerabilities, weaknesses, and compliance violations earlier in the process. It builds on DevOps "first principles" to add security to the development process seamlessly.



How Apiiro Can Help

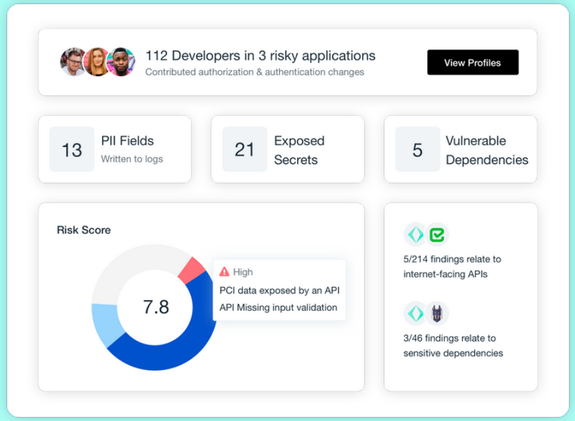
Apiiro helps you build a risk-based & measurable Application Security program. Our Application Risk Management platform provides you with complete risk visibility and control, from design to code to cloud. Our multidimensional approach to application security will help accelerate application delivery while lowering costs and reducing risk.

Apiiro is re-inventing the secure development lifecycle for agile & cloud-native development. Apiiro can help you:

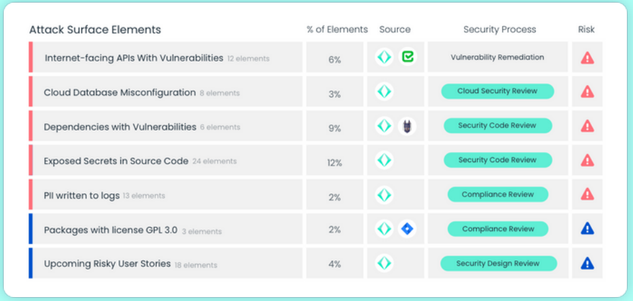
- **Define Success** with risk-based metrics that help you measure your AppSec program at both business & technical levels
- **Gain Risk-Based Visibility** with a Real-Time Application Inventory & Asset Discovery
- **Remediate the Risks that Matter** with a risk-based Remediation Work Plan
- **Automate Code Governance** with detailed workflows and a flexible Code Governance Engine
- **Approach the SSDLC Holistically** with a Risk Dashboard that covers all SSDLC tools and processes
- **Shift Left & Extend Right** with Security Champion identification and the context to trigger context-sensitive developer training

Illustration 7

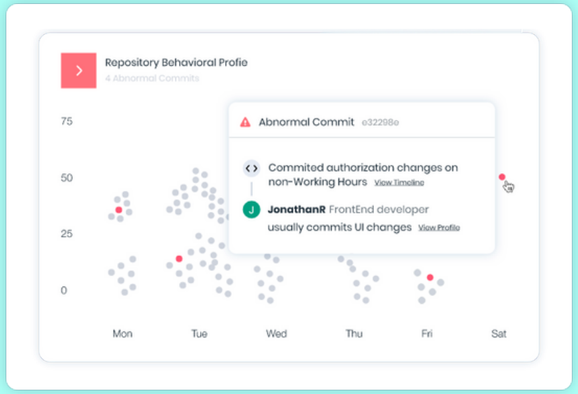
Apiiro solutions



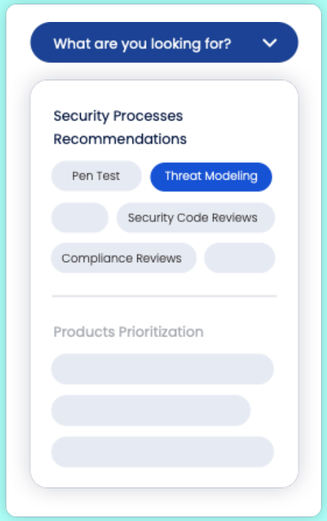
Application Inventory & Asset Discovery



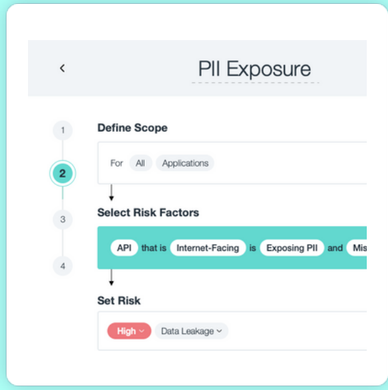
Risk Assessment & Change Management



Git & CI/CD Security & Integrity



SSDLC Processes & Tools Orchestration



Security & Compliance Assurance

How Apiiro Can Help

Apiiro gives you a 360° view of security & compliance risks across applications, infrastructure & open-source code, developer experience, & business impact. We are pioneering the concept of “Multi-Dimensional Application Risk Analysis" in order to identify risky material changes and help you focus on remediating the most critical code risks to your organization.

Our Code Risk Engine™ analyzes risk with context by learning all application code & Infrastructure-as-Code, combined with data from API gateways, communication & Security tools, ticketing systems & findings from pen tests and compliance reviews.

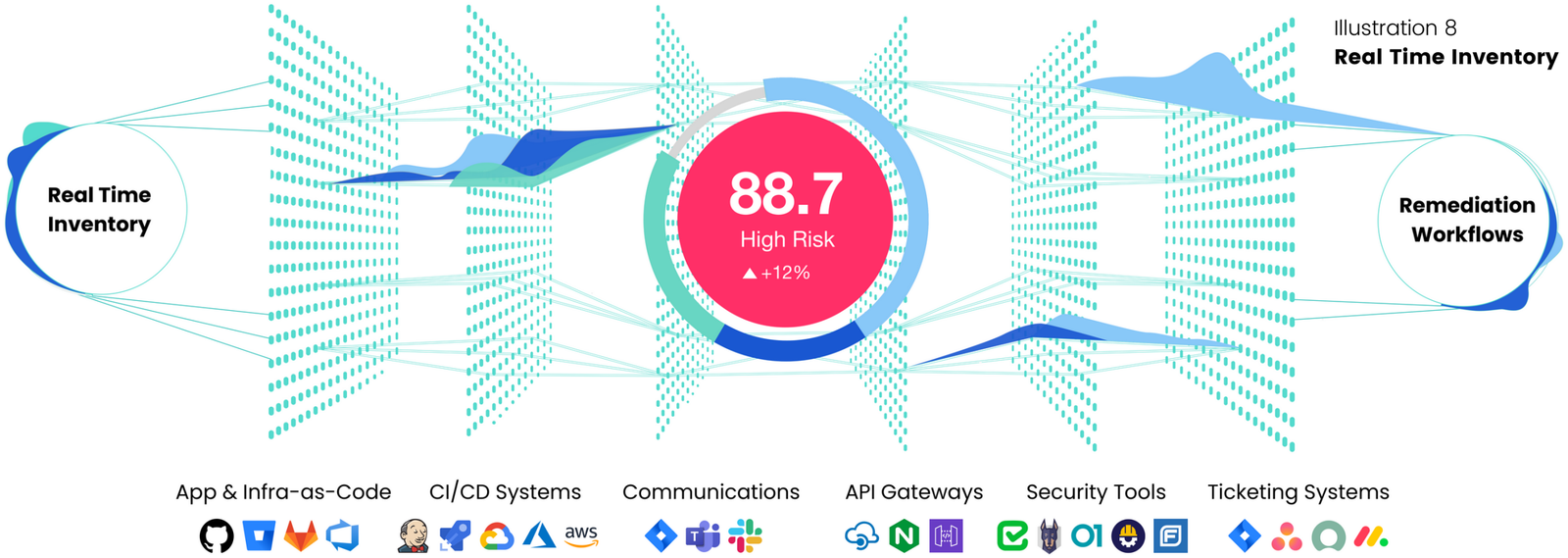


Illustration 8
Real Time Inventory

With Apiiro, you can build a mature Application Security program that enables you to confidently deliver both faster and more securely.

To learn more, schedule a demo today!

Conclusion

To stay relevant, Security professionals need to up-level their thinking and messaging to better align with how executives make decisions. The way to do this is to move towards a risk-based Application Security program. By doing this, security practitioners can better participate in business discussions that come down from the executives and Board - this will drive a true digital transformation.

The benefits of this approach will help stakeholders across the company:

CIOs and CISOs Executives will gain a high-level & contextual risk into the real risks to the business. They will be able to speak the same language when discussing security and risk.

Security Architects and AppSec Leaders

A risk-based approach to DevSecOps will fundamentally change the role of Security Architects/AppSec Engineers in the organization. Instead of chasing, investigating, and triaging vulnerability scan results, Security Architects can spend more of their time on strategic issues and continuous learning. Their role will shift to a quest for knowledge; to understand the latest security and compliance concerns, secure coding techniques, and defensive practices. They will then spend more time teaching developers and disseminating their knowledge than chasing vulnerabilities, weaknesses, and compliance violations. That knowledge transfer will become an essential part of DevSecOps and a new avenue of collaboration between security and developers.

Developers Empowering developers by giving them the right context and information at the right time is the only way to foundationally change how we do software development. This is a big change that requires buy-in from multiple areas across the organization, but the ones that do this successfully will be able to release code both faster and more securely - and have happier, more-knowledgeable, and better trained software development teams while they do it.

Conclusion

Penetration testers Receiving contextual alerts related to risky material code changes will allow Pen Testers to perform incremental pen-tests on only the changes that matter. They won't need to rely on developer surveys or fixed schedules to know when to perform. In addition, Pen Testers will be able to focus their efforts better than ever, focusing more time on risky code areas & ultimately identifying more security issues.

Legal/Compliance **Lawyers and Compliance experts** can easily and accurately identify compliance issues, such as open-source software licenses, copyrights, etc.

Developing a risk-based Application Security program will change your entire approach to security in software development. It will help you stop thinking about security as a checkbox function and more of a context-aware process. You'll accelerate delivery and finally achieve true Digital Transformation.