VERACODE

# AppSec Best Practices vs. Practicality

## What to Strive for and Where to Start

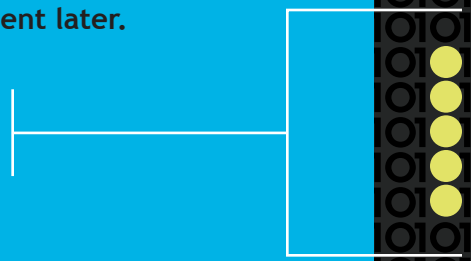
01

# 01

In a perfect world, you would use nothing but best practices to guide your AppSec program's development and implementation. In the real world, however, you're limited by time, budget, staff expertise, culture, and executive support. Fortunately, those limitations don't have to stop you from creating an AppSec program that can still get powerful results.

When building out your AppSec program, remember that something is always better than nothing. While achieving best practices should be your end goal, even a few practical steps taken now can help craft an AppSec foundation that can move the needle while positioning your program for improvement later.

In this guide, we'll look at five key AppSec best practices and discuss the practical steps you can take immediately to get the ball rolling in each of these areas.



BEST PRACTICE 1

# Use More Than One AppSec Testing Type

You wouldn't get your vision tested if you had a toothache or be tested for diabetes if you had a runny nose. At the same time, you wouldn't just get a simple cholesterol test and assume your annual checkup was complete. Just like with your personal health, your software security health requires a variety of tests that examine different things in different ways. For effective AppSec, you must use the right test at the right time.

Each testing type has its own strengths and weaknesses, with no one tool able to do it all. You need to use the different advantages of multiple analysis techniques if you hope to achieve comprehensive application security.

By using a combination of static analysis, dynamic analysis, software composition analysis, interactive application security testing, and penetration testing, you can drive down risk across the entire application lifetime from development to testing to production.

# Application Security Testing Types and Their Capabilities

**Each testing type provides a different area of focus, along with varying speeds and costs.**

| Capabilities | Static Analysis | Dynamic Analysis | Software Composition Analysis | Interactive Application Security Testing | Manual Penetration Testing |
|---|---|---|---|---|---|
| Flaws in Custom Web Apps (CWEs) | ✓ | ✓ | | ✓ | ✓ |
| Flaws in Custom Non-Web Apps (CWEs) | ✓ | | | ✓ | ✓ |
| Flaws in Custom Mobile Apps (CWEs) | ✓ | | | ✓ | ✓ |
| Known Vulnerabilities in Open Source Components (CVEs) | | | ✓ | | ✓ |
| Behavioral Issues (CWEs) | ✓ | | | ✓ | ✓ |
| Configuration Errors (CWEs) | | ✓ | | | ✓ |
| DOM-Based Cross-Site Scripting | ✓ | ✓ | | | ✓ |
| Business Logic Flaws (CWEs) | | | | | ✓ |
| Coverage of Full Application | ✓ | ✓ | ✓ | | ✓ |
| Repeatable Process for Automation | ✓ | ✓ | ✓ | ✓ | |
| Scalable to All Corporate Applications | ✓ | ✓ | ✓ | ✓ | |
| Scan Speed | Seconds to Hours | Hours | Seconds to Minutes | Seconds to Minutes | Days to Weeks |
| Cost | $$ | $ | $ | $ $ $ | $ $ $ $ |

WHAT'S PRACTICAL

# Start With What Makes the Most Sense, Then Add More Later

Take the time to develop an AppSec strategy that helps you determine where you need to add AppSec testing the most. You can then evaluate each technique to find the one that can have the most immediate impact for the lowest cost and in the least amount of time. Once you become comfortable with that testing type, you can begin to incorporate additional testing types as necessary and as budget allows.

While some tools are more foundational than others, there is no single tool every company should start with. The technique that's right for you will depend on the following factors:

## Your release cadence

If you release once a quarter, manually kicking off a scan may work fine for you. If you're working in a CI/CD fashion with several releases a day, this wouldn't be feasible. In this case, make sure you fully automate scanning with each code commit. Faster scan types, like static analysis, can provide immediate feedback with each commit; slower scan types, like dynamic analysis, may have to be run once a day.

## Your risk tolerance

Not all apps are created equal. The app you create to display a cafeteria menu requires a different level of security than that of an online banking app. The more important it is that an app is secure, the more you should look at in-depth tests like penetration testing. Weigh risk tolerance vs. time to market: If time to market is more important for your app, you may be comfortable releasing it with a medium-severity vulnerability and fixing it within a week. If your app is critical, you should fail the build and halt deployment until it's resolved.

## Programming languages

If you're using a less-mainstream language, you may find that there are limitations to the tests you can conduct. Some testing types, such as dynamic analysis and penetration testing, are less specific to programming languages than static analysis, software composition analysis, and interactive application security testing.

## The relationship between security and development

Security teams that don't have access to source code may have to use tests like dynamic analysis. As they prove the value of testing, it can open the conversation to incorporating static analysis that can help find flaws earlier in the development cycle.

## Where your most significant risks are

If you rely on open source libraries, you may be at risk for vulnerabilities in code you didn't even write. In this case, you might prioritize software composition analysis to track vulnerabilities in third-party code.

## Executive buy-in and AppSec budget

No security team has a blank checkbook. Once you determine your security testing needs, you need to make the case to company leaders for adequate funding. Often, you can gain executive buy-in by attaching your security testing initiative to larger enterprise initiatives like Agile, DevOps, and digital transformation.

## AppSec maturity

Every enterprise is at a different place in its AppSec journey. If you're just starting out, your goal should be to incorporate one or two testing methods. Once you get to that point, you can then look for ways to incorporate additional techniques until you have the right mix for your organization and for each application, based on risk level and underlying technology.

**The reduction in the median time it takes to fix security flaws** for those who scan their apps for security

12 times or less per year

68 days

260+ times per year

19 days

**That's a 72% reduction.**

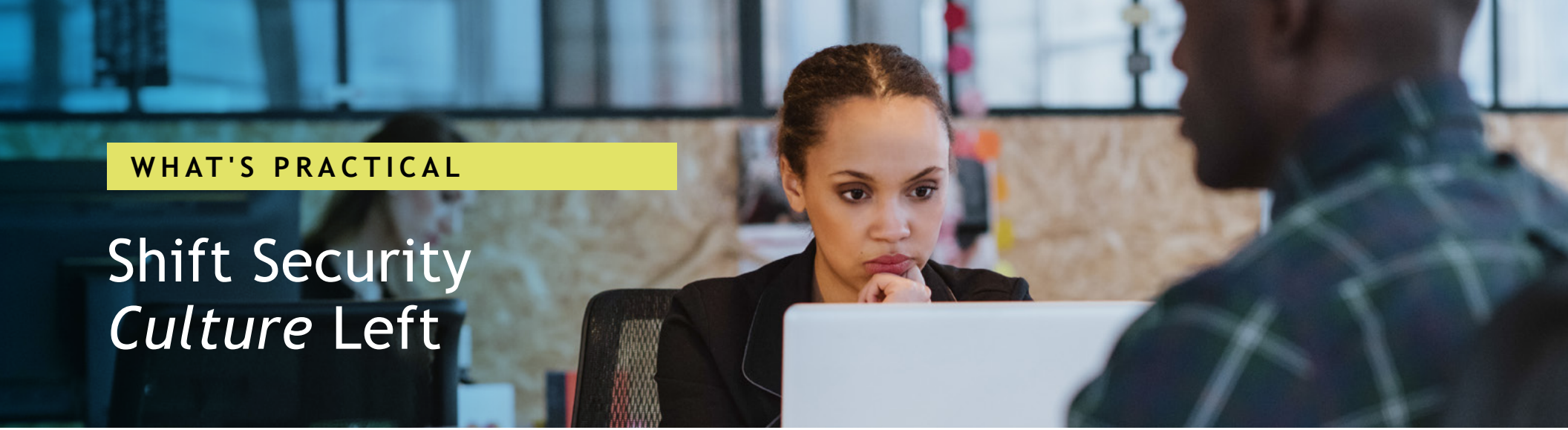
Source: ***State of Software Security Volume X.***

BEST PRACTICE 2

# Shift Security Left

Security shouldn't be a step — it should be a state of mind. That's because today's expectations for on-time delivery of high-quality software means enterprises simply don't have the time to wait until the end of the development cycle to start testing security.

As enterprises continue to move from yearly product releases to monthly, weekly, or even daily release cycles, security needs to shift left so that developers are testing security as they work. By moving security testing into the realm of the developer, security testing becomes faster, easier, more effective, and less costly.



WHAT'S PRACTICAL

# Shift Security *Culture* Left

"Shifting security left" is one of those popular industry catchphrases that's unfortunately easier said than done. To do so takes more than just technology. It takes an honest assessment of the culture of your organization to figure out how to give your development team more ownership of security.

Here are three things every organization can do to make development more security-minded:

## Understand how development works

Every development team has its own unique tools and processes. Before you prescribe solutions that might disrupt their work, take the time to understand how your development team is building software so you can find opportunities to organically integrate security testing into the process.

## Automate, automate, automate

By using APIs to integrate automated security tools into the CI/CD pipeline, you can incorporate testing without the need to hand off code to another team. This makes it simpler for developers to fix issues in the moment instead of weeks later when they get a report back from the security team.

A good place to start when looking to automate security testing is to automatically kick off static analysis scans in your build process.

## Shift knowledge left

The more you can embed security knowledge in the development team, the more likely they'll deliver secure code from the beginning. A good first step: At management level, make security a shared goal between security and development. Next, appoint and start educating security champions (more on that on page 16).

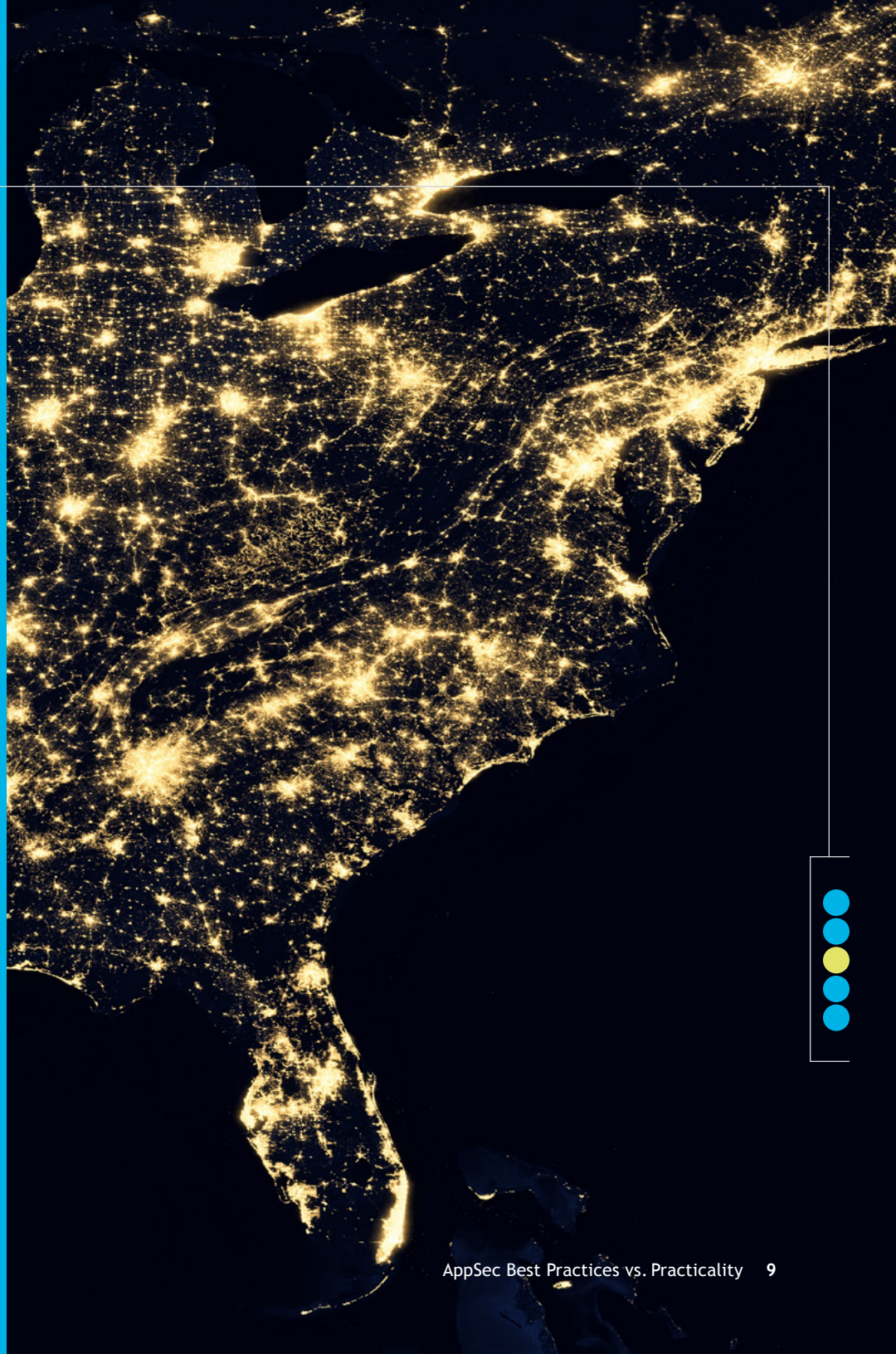
BEST PRACTICE 3

# Fix Everything Fast

Finding flaws faster in the development cycle is all well and good. But a list of flaws to fix does you no good unless you fix them. Crucial to an effective AppSec program is not just the ability to discover flaws earlier in development, but to fix them quickly once they're discovered.

By building and integrating vulnerability testing into their CI/CD pipeline, some development teams have been able to reduce mean time to remediation by 90 percent, with resolutions reduced to 15 minutes down from 2.5 hours on average.[1] The faster you close vulnerabilities, the more flaws you can fix, which means the less risk your apps will face over time.

[1] The Total Economic Impact of the Veracode Application Security Platform, Forrester, March 2019.



WHAT'S PRACTICAL

# Prioritize Fixes While Creating Fewer Vulnerabilities

Life is about priorities. If the timer for the cookies you have in the oven dings at the same time as the timer for your dryer, you wouldn't check to see if your socks are dry first. Just as you'd prioritize the cookies to keep them from getting burnt, you need to prioritize the application flaws that place you at the most risk. Then, once your high-priority flaws are fixed, you can then go down the list to the next most-critical flaws.

Of course, the easiest flaws to fix are the ones that don't exist in the first place. That means you should invest just as much in tools that detect flaws before they make it into your code as you do in tools that detect flaws later in the development cycle. Here are three ways you can begin to improve your remediation speed:

## Scan more frequently

In our *State of Software Security Volume 10*, we found that organizations that scan the most often (more than 260 times per year) reduced their median fix time by 72% while tripling their fix rates over teams that scan infrequently. The more you scan, the quicker and easier you'll be able to fix small flaws while they're still small.

## Prioritize your fixes

Revisit your AppSec policy to ensure you're prioritizing flaws not just by defect severity, but by other risk factors such as the criticality of the application or defect exploitability. In addition, make sure you include both newly found and older vulnerabilities in every remediation sprint so that you don't build up flaw debt.

## Reduce new vulnerabilities

The fewer flaws you have in the first place, the fewer flaws you'll need to fix later on. By integrating automated security tools early into the development life cycle, training developers on secure coding practices, and providing guidance for fixing security-related defects, you'll be able to significantly increase your organization's fix rate early on.

BEST PRACTICE 4

# Embed Security Champions on Each Development Team

There's a critical security skills gap in the development industry that has put developers with adequate security experience in short supply. In fact, a recent Veracode/DevOps.com survey showed that nearly 40% of respondents say they struggle to find all-purpose DevOps gurus with sufficient knowledge of security testing. Not only that, but 76% of developer survey respondents who attended college reported that they weren't required to complete any security courses while in school.

By embedding security champions within your development teams, you'll be able to put security front and center in the development process and help amplify the security message at the team level, saving valuable time later on. A security champion isn't a security expert, but a developer or product team member who can keep an eye out for security issues, acting as the security conscience of the team.

WHAT'S PRACTICAL

# Build Up Your Security Champions' Capabilities

You can't just choose a random developer to be your security champion and call it a day. You have to make a long-term investment to ensure they have the knowledge and support they need to effectively champion security over the long term. Here are a few first steps to start incorporating security champions in your organization:

## Define the program

Make sure leadership, security, and development team leaders are all willing to invest the time, money, and resources needed to make security champions successful. This may include making changes to traditional security practices, implementing automated tools, and ensuring security champions get the extra training they need to be effective.

## Build a security culture

Security needs to make sure it builds relationships with development teams so they buy into the idea of security champions. By making it easy for people throughout the organization to learn about security, it will help them get invested and want to be a part of the program.

## Identify your champions

As you build relationships with your developers, you're likely to find people who are naturally interested in security. You can also ask your development managers if they have any team members they'd like to nominate. Since being a security champion is extra work, try to find volunteers who are passionate about the topic and want to learn, instead of assigning the task to someone who might not be as invested.

## Nurture your champions

Don't expect your volunteers to have any prior security expertise. For the first phase of your program, the goal is to have your security champion simply be able to identify whether a member of the security team needs to get involved. To do so, you'll want to give the champion additional security training beyond what the rest of the team receives so they have a higher awareness of what to look for. As they become more familiar with security concepts, they'll be able to do more advanced tasks, such as code reviews.

BEST PRACTICE 5

# Measure Your AppSec Results

At the end of the day, your AppSec program is only as good as the results it helps you achieve. And the only way to know that is with metrics. But identifying the right metrics is only the beginning. You have to also present them to decision-makers in a way that makes it easy for them to understand and take action. This requires devising a program that supplies the right metrics to the right people at the right time.


WHAT'S PRACTICAL

# Focus on Your Policy Metric

The good news: Executives love data. The bad news: Your executives are already drowning in a sea of data. That means you need to be thoughtful about making your AppSec performance simple to measure and understand if you want them to pay attention. By focusing on your policy metric first, you make it easy for development teams, security, and leaders to make the tradeoff decisions that need to be made between fixing vulnerabilities and accepting risk in a way that's consistent with the goals of the business.

## Start with the policy metric

The policy metric is the percentage of applications you're scanning that are passing your compliance policy. You may choose to measure overall compliance or focus on your most high-risk application categories.

- **Work with an achievable policy.** To improve your policy metric, develop a security policy that makes sense to your development team. If your policy is too strict or difficult to understand, developers are likely to work around it in order to get the application into production instead of following the policy.
- **Set a baseline.** Don't make your security policy a moving target. Create a baseline that mitigates your most pressing security issues. Once that becomes the norm, you can then add incremental improvements that seek to reduce additional flaws.
- **Rank your risks.** Apps that are accessible on a website or that process PII can cause more damage than internal-facing apps. Make sure you prioritize high-ranking risks, not just easy-to-protect risks, when defining your policy metric.

## Keep your goals front and center

Whether it's development speed, customer satisfaction, or innovation, your business goals should inform your AppSec goals, which should in turn inform how you define your policy metric.

## Don't forget to tell the story

Once you have your policy metric in place, don't just start throwing numbers around. Make sure you provide context behind the number to help company leaders understand what the data is trying to tell them.

## Think beyond policy metrics

Once you've mastered your policy metric, you can branch out into other metrics that paint a richer picture. Your time-to-resolution and your flaw prevalence are both valuable metrics that can help you focus your resources.

"There is no application security silver bullet. It's going to take more than one automated technique and manual processes to secure your applications."

— **Chris Wysopal,** Veracode co-founder and CTO

**Organizations that scan their code for security**

{ >300 times per year

have 5 times less security debt

than less frequent scanners.

Source: ***State of Software Security Volume X.***

# Practical Today, Best Practices Tomorrow

AppSec excellence is a journey, not a destination. In fact, achieving AppSec best practices on Day One is all but impossible. Instead, take incremental, concrete, and practical steps today to build a foundation of AppSec success. While achieving best practices might feel overwhelming at first, by breaking it into small steps you'll be able to build the culture, toolkit, and processes to achieve AppSec maturity sooner than you think.

For more details on AppSec best practices and the practical first steps you can take to develop your AppSec program, check out the sessions in our virtual summit, Your AppSec Game Plan.


VERACODE

Veracode gives companies a comprehensive and accurate view of software security defects so they can create secure software, and ensure the software they are buying or downloading is free of vulnerabilities. As a result, companies using Veracode are free to boldly innovate, explore, discover, and change the world.

With its combination of automation, integrations, process, and speed, Veracode helps companies make security a seamless part of the development process. This allows them to both find and fix security defects so that they can use software to achieve their missions.

Veracode serves more than 2,000 customers worldwide across a wide range of industries. The Veracode Platform has assessed more than 8 trillion lines of code and helped companies fix more than 36 million security flaws.

Learn more at **www.veracode.com**, on the Veracode **blog** and on **Twitter**.

Copyright © 2020 Veracode, Inc. All rights reserved. All other brand names, product names, or trademarks belong to their respective holders.