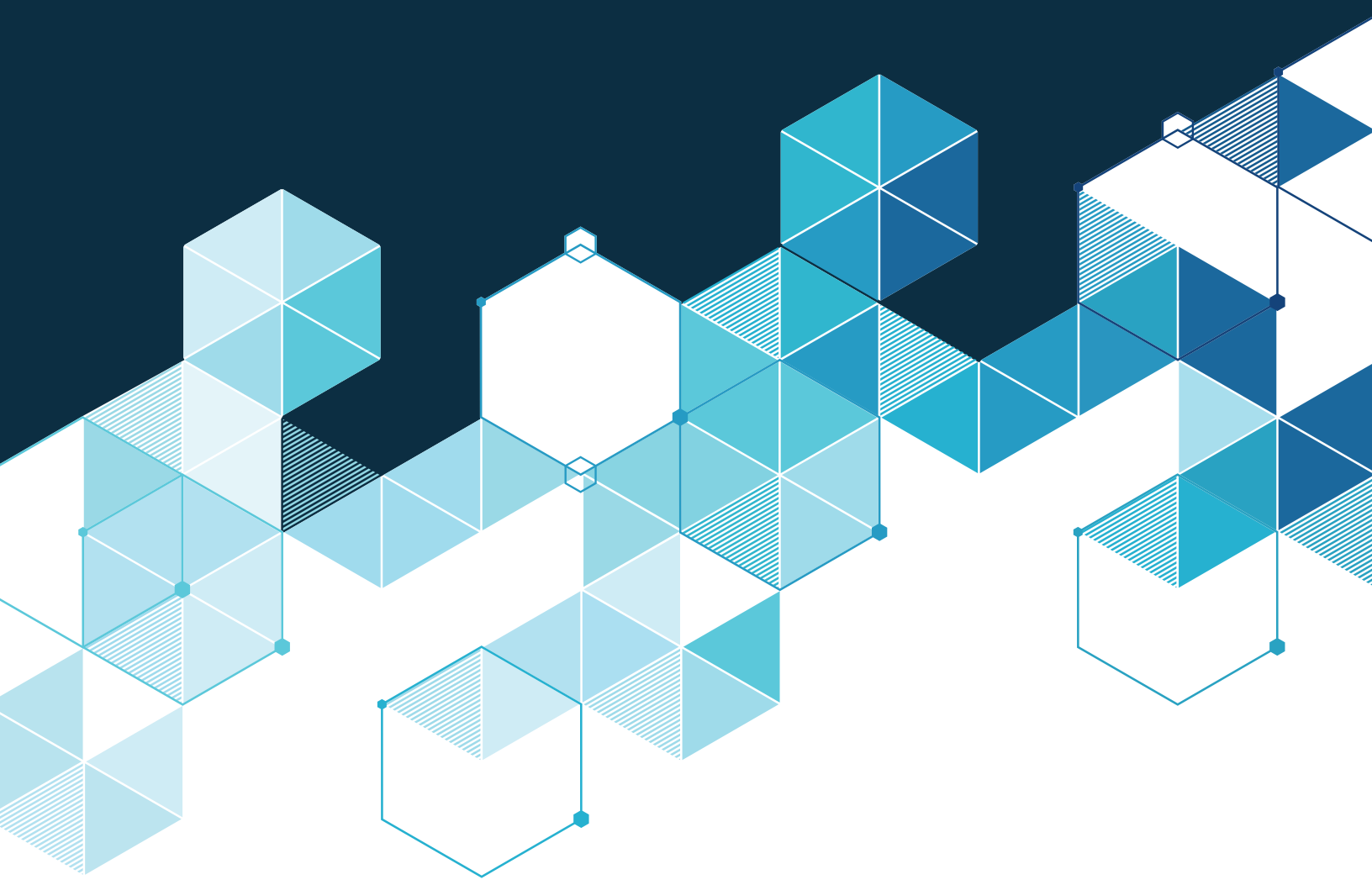
STATE OF SOFTWARE SECURITY

DEVELOPER GUIDE



ca technologies | VERACODE

THE STATE OF SOFTWARE SECURITY TODAY

What It Means for Developers

In today's digital economy, developers play a vital role in translating business needs into functional applications that drive revenue and improve processes. To do their jobs well, developers obviously must be able to deliver features at the rapid pace of changing business needs. Applications need to work. Less obviously, but just as important, developers have to deliver code with as little business risk as possible.

If a new application meets functionality requirements but causes a security breach in the process, the advantages to the business of that application may be outweighed by the costs. You can't call your code or application great if it isn't secure. That's why the responsibility for software security has shifted to developers at many organizations. On a practical level, this means that, as a developer, you are shouldering more daily responsibility for testing and remediation of security vulnerabilities as code makes its way through the software development lifecycle (SDLC). For this reason, developers must be empowered to truly own application security as a function of overall application quality.

Now, the security team still offers a high degree of strategic support and consultative insight to help the process along. But in today's DevOps and Agile environments, developers can no longer throw an untested application over the wall to security and expect the security team to take care of hardening it. Smart developers recognize that establishing a proactive stance on application security starts with knowledge. You know that you have to bring the same kind of thirst for knowledge about security best practices as you do for learning new languages, frameworks, and other tools.

Developers are shouldering more responsibility for security testing and remediation of vulnerabilities.

The State of Software Security Report

CA Veracode has always endeavored to be a partner for both the security community and the developer community in improving your AppSec know-how. As part of that mission, we offer our annual State of Software Security (SOSS) report, which analyzes the wealth of data we gather from our platform. The most recent SOSS report is based on data from 400,000 application scans over a 12-month period, from April 1, 2016 to March 31, 2017, covering applications written in more than a dozen programming languages, across organizations large and small, and in a broad swath of industries.

In addition to the SOSS report, we're publishing this supplemental report for developers, with the aim of helping you take away practical knowledge based on feedback from our application scans. The State of Software Security Developer Guide provides data-driven insights especially for the developer community. We encourage you to dig into this report, as well as the SOSS report. All this rich data can help you benchmark your own performance and better understand the challenges and successes of creating great, secure software.

CONTENTS

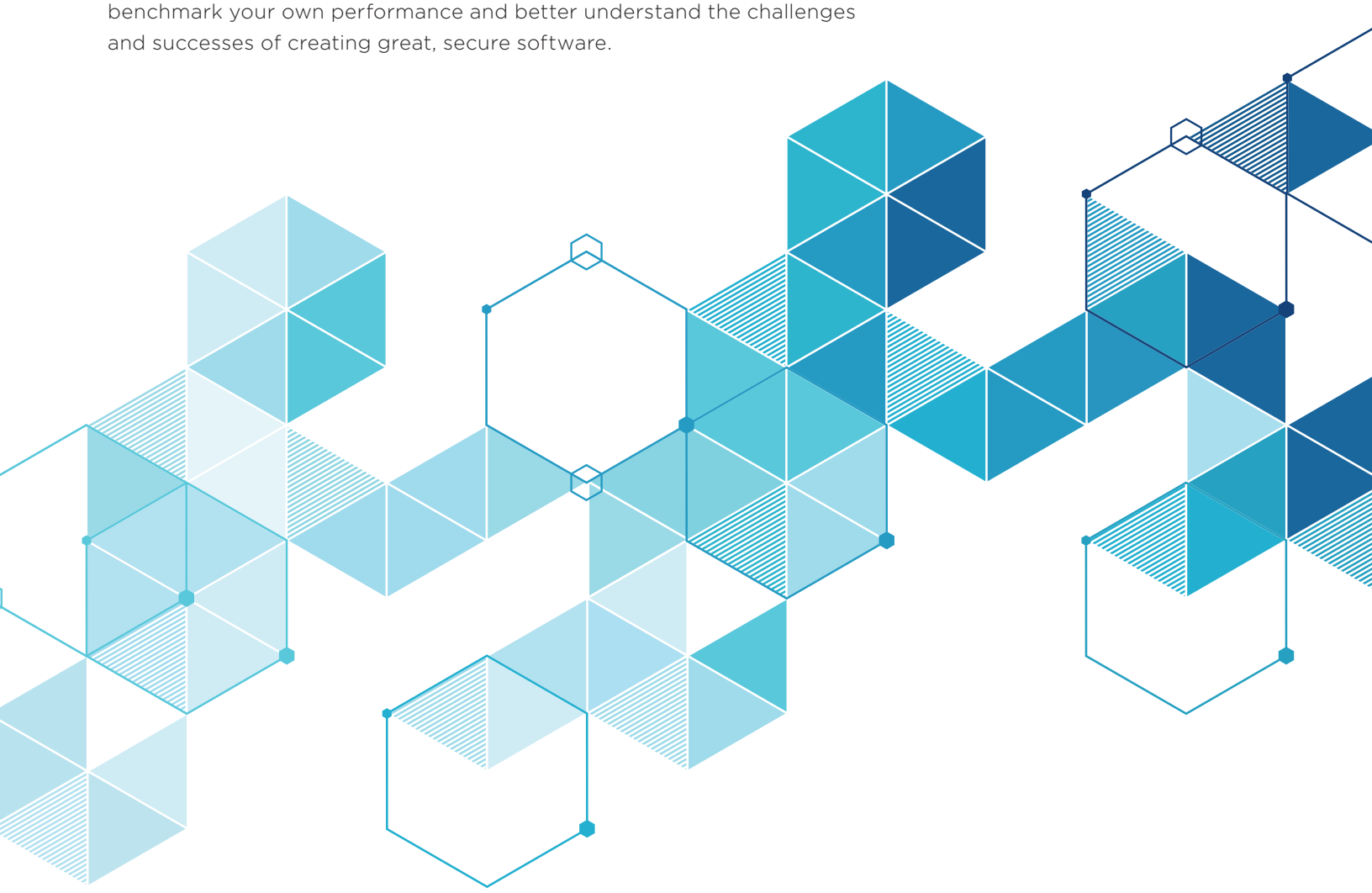
2 WHAT IT MEANS FOR DEVELOPERS

4 YES, DEVELOPERS CARE ABOUT SECURITY

5 TOP TAKEAWAYS FOR DEVELOPERS

8 BEST PRACTICES FOR RAISING YOUR APPSEC PERFORMANCE

18 CONCLUSION



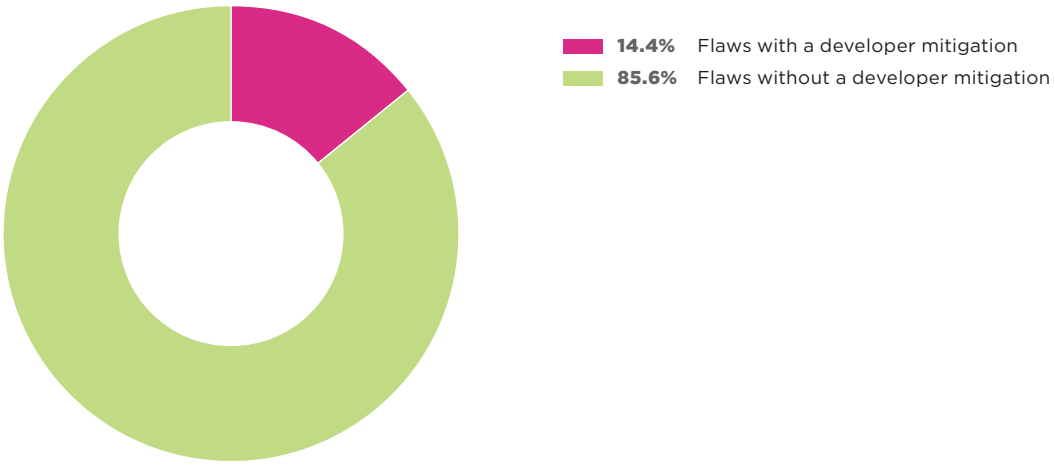
Yes, Developers Care About Security

While many security firms and security practitioners tend to throw developers under the bus, CA Veracode understands the reality you face in your job. You're expected to meet demands from bosses who don't always put security at the top of the requirements list, and you're dealing with a shortage of resources for AppSec education. But that doesn't mean you don't care about security.

One way we examined developer attitudes towards AppSec was to look at stats around mitigation rates. In the CA Veracode platform, developers are given the choice of either looking at the code and learning how to fix the issue, or documenting a mitigation — in other words, writing down a reason why you're not going to fix the issue right away. For example, if there are some other compensating controls that lessen the risk of a particular vulnerability, the developer would document that as a mitigating factor and move on.

The interesting thing here is that, for the most part, developers don't try to game the system by rejecting findings primarily as false positives, or as mitigated by design. In fact, in the past year, developers documented mitigations for just 14.4% of all the flaws found by our platform.

DEVELOPER MITIGATIONS OUT OF TOTAL FLAWS FOUND

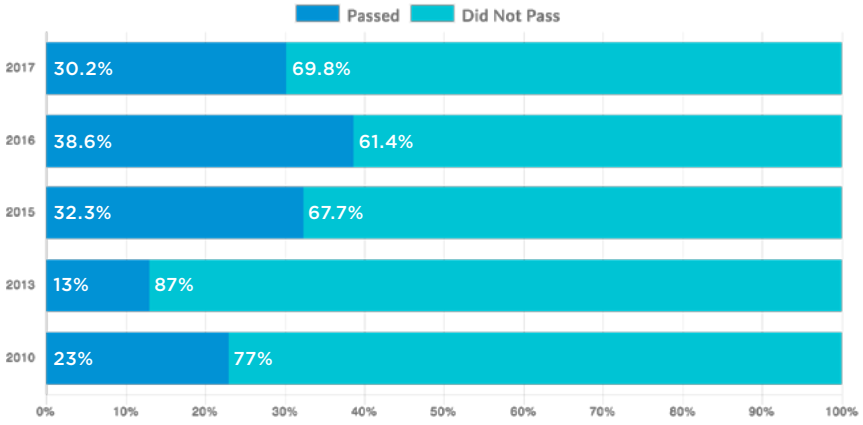


Top Takeaways for Developers

Despite signs that developers are taking security seriously, the overall security picture is more mixed. As things stand, when organizations don't operate with formalized AppSec processes in place, the chances that an application will pass muster with security standards are slim.

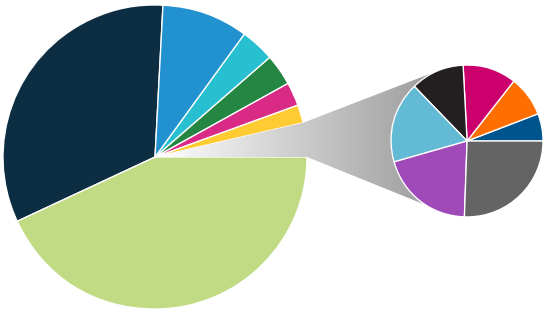
OWASP TOP 10 POLICY PASS RATE

Percentage of Applications Passing on First Scan



- Our figures show that applications passed OWASP Top 10 policy only 30% of the time last year, when first scanned by our platform. In fact, the OWASP Top 10 pass rate has consistently been below one-third of applications for the past five SOSS reports.

BREAKDOWN OF LANGUAGES SCANNED

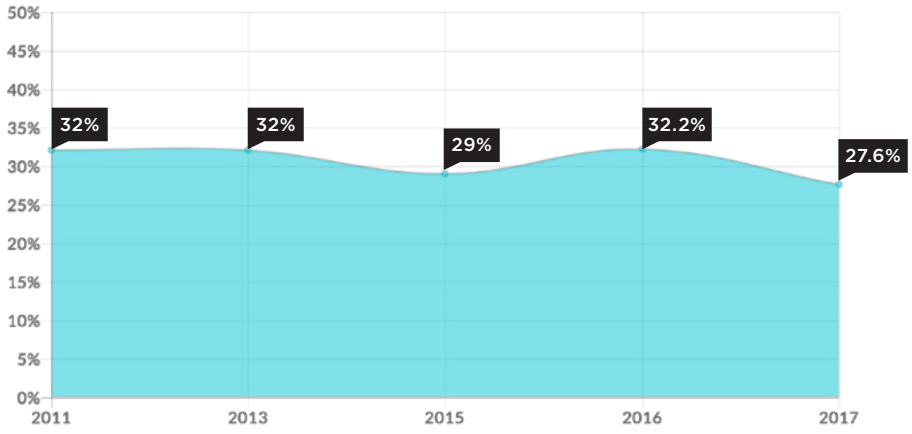


43.2%	Java
32.7%	.Net
9.3%	JavaScript
3.5%	Android
3.3%	PHP
2.5%	C++
2.0%	iOS
0.9%	iOS Swift
0.7%	VB6
0.6%	COBOL
0.4%	Python
0.4%	ASP
0.3%	ColdFusion
0.2%	Ruby

- For point of reference, this chart shows the breakdown of applications we scanned in the past year by development language, with a heavy weighting on Java and .NET.

SQL INJECTION TREND

Percentage of Applications Affected



As we explained in the SOSS 2017 report, many of the same types of vulnerabilities keep cropping up at approximately the same rate, year in and year out, when new applications are first scanned. For example, SQL injection flaws appeared in just under 28% of newly scanned apps in 2017, and were detected in around one-third of applications for the past five SOSS reports.

PREVALENCE OF MAJOR VULNERABILITY CATEGORIES BY LANGUAGE

Percentage of Applications Affected; *Static analysis on initial scan, policy scans only*

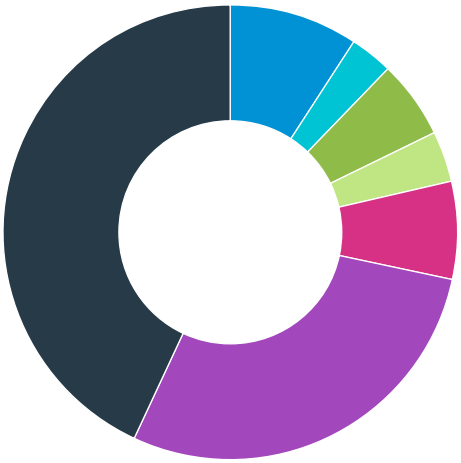
Language	Cross-site Scripting	Credentials Management	SQL Injection	Cryptographic Issues
.Net	14%	20%	31%	16%
Android	31%		61%	20%
ASP	10%	24%	10%	18%
C++			26%	
COBOL	4%			4%
ColdFusion	3%	38%	3%	21%
iOS	24%		45%	
iOSSwift	21%		35%	
Java	27%	23%	30%	14%
JavaScript	15%	21%	13%	9%
Perl*		20%	40%	
PHP	43%	51%	47%	32%
Python	5%	19%	25%	6%
Ruby*		3%	33%	15%
VB6	34%		18%	19%
OVERALL	38.2%	40.4%	26.6%	57.6%

*** Small sample size**

When looking at the prevalence of major vulnerability categories, no language goes unscathed. All of the languages show a statistically significant incidence on first scan for some of the most highly impactful types of vulnerabilities. These numbers show that when developers operate without any influence from application security testing, flaws are not only inevitable – they're inevitable in high numbers.

- High rate of application risk
- Moderate rate of application risk
- Low rate of application risk
- Not scanned for vulnerability type

DAYS TO CLOSE FLAWS

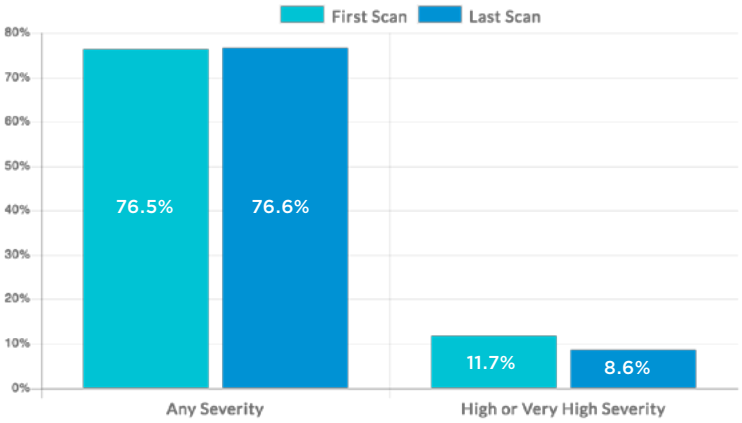


9.1%	Less than 8
3.1%	8 to 14
5.5%	15 to 30
3.6%	31 to 45
7.9%	46 to 90
28.3%	More than 90
42.4%	Still open at the end of measurement period

Even once AppSec programs are put into place, it takes meaningful engagement across software engineering, security, and operations teams to make big improvements. Last year, we found that fewer than one-third of flaws (29%) were closed in 90 days or less, and 42% of flaws were never closed during the measurement period.

APPLICATIONS WITH AT LEAST ONE VULNERABILITY

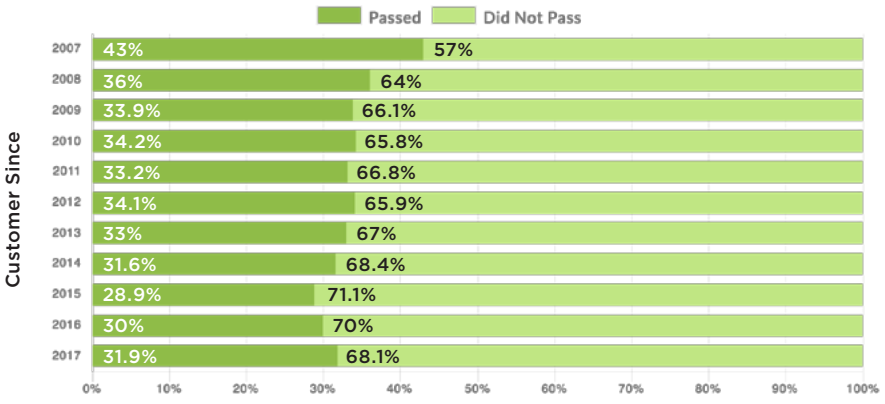
Percentage of Applications



Nevertheless, organizations make progress where it counts the most. While about 77% of applications have at least one vulnerability on both first scan and last scan, organizations are making headway in reducing the number of applications with the highest severity flaws. Between the first security test and the latest scan in our measurement period, the rate of applications with high and very high severity vulnerabilities declined by 26%.

APPLICATION SECURITY PROGRAM MATURITY

OWASP Top 10 Pass Rate (All Time)



What's more, the longer that developers and security teams work together to programmatically improve application security, the better they get at it. Organizations with AppSec programs in place for 10 years have an all-time OWASP Top 10 pass rate 35% higher than those with programs in place for a year or less.

Best Practices for Raising Your AppSec Performance

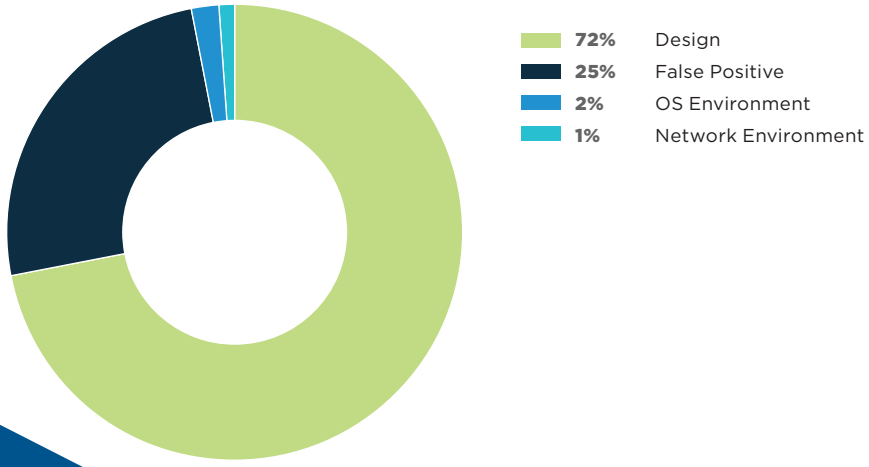
Long-term programmatic success in application security requires a deep commitment from all stakeholders: business executives, IT leadership, and security advisors among them. But perhaps no group is as important as developers. You're closest to the code, and you're one of the stakeholders poised to make the biggest difference in AppSec — if you're given the right tools and resources to make it happen.

So what does it take for developers to start chipping away and making a difference? Based on our analysis of the data, we think there are five big things developers can do to greatly accelerate progress.

Think Like an Attacker

We already touched on mitigation documentation as proof that developers don't brush off security. Overall, the rate at which developers use mitigation shows a commitment to security. Fewer than 15% of developers question findings with documented mitigations. And of that fraction, only 25% of documented mitigations are reported by developers as false positives.

DEVELOPER MITIGATION REASONS



Among the other 75% of the documented mitigations, developers argue that something in the OS environment, network environment, or the design of the application itself reduces the risk of whatever vulnerability the scan uncovered. In many cases, these explanations are likely legitimate. For example, perhaps the developer isn't using a secure cryptographic algorithm but the value being encrypted is not data of any value to the business or to an attacker. Other times, insecure cryptography is probably less defensible.

However, as we analyzed mitigation documentation, we found that some developers may be brushing off security recommendations based on some unsound assumptions about how applications can potentially be attacked. Because these are free-form documentations, it's difficult to quantify mitigation comments. However, we can provide some anecdotal examples of developers making unsound interpretations of scan results.

Often developers have disagreements with static analysis based on different assumptions about the trustworthiness of their inputs, such as databases and file stores. Other common points of friction include whether input from a third-party component may be trusted, and whether functionality designed to be internal can be attacked. One common risk factor we found in mitigation comments was that some developers are still trusting inputs from users to a troubling degree. While 99% of legitimate users would never enter anything malicious into an input field, it's that slim minority of attackers we still need to worry about.

EXAMPLES OF QUESTIONABLE DOCUMENTED MITIGATION REASONS

"Injection source has been verified and it cannot be attack via SQL injection"

"Development team has assured us that all user input is sanitized before being used. Secondly, if there is a scenario to use a value directly, it is only for internal purposes and cannot be tampered externally. With this being the case we have agreed to mitigate these flaws by design."

"The values are coming from the GUI page as a list of select boxes which cannot be modified."

"This type of flaw is impossible to occur in line 1 of a Java class."

"This is controlled by internal logic — the user cannot go to a page we do not allow."

Misunderstanding how applications may be attacked sometimes leads developers to reject findings that are valid, including mitigation comments that "Nobody would ever modify that." We see this in other domains, not just in static analysis. But the fact remains that sometimes developers think primarily from the perspective of a legitimate user, rather than thinking as an attacker would. The point is that developers need to think not only in terms of use cases, but also abuse cases. You need to be absolutely certain that inputs can be trusted before explaining away potentially legitimate flaw findings.

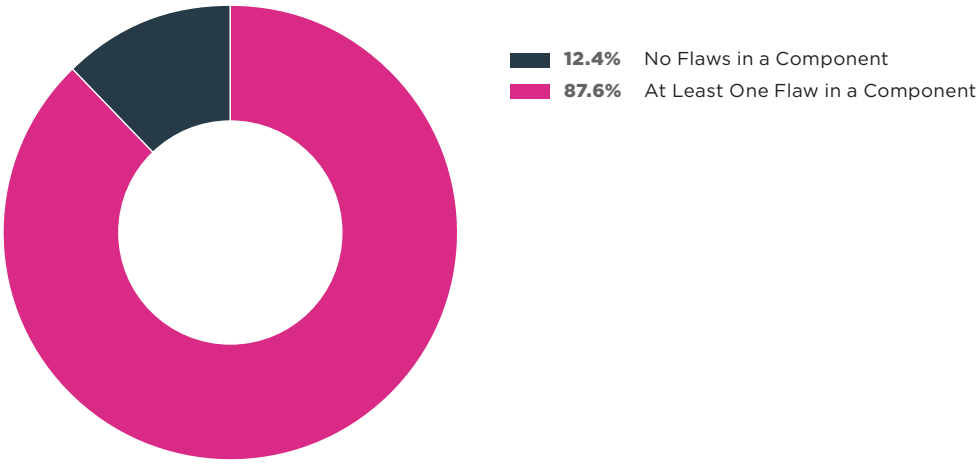
Bring Greater Discipline to Component Use

We are beginning to face a big crisis in securing the software supply chain. As development processes become atomized through microservices, a lot of development velocity today hinges on how well an organization can stitch together open source components with their own code. Modern developers understand that integrating components rather than developing a monolithic code base will help them innovate faster and with less technical debt.

The downside to all this acceleration is that open source components often carry their own set of security vulnerabilities and risks. Unfortunately, many organizations either aren't aware of them or choose to ignore them in favor of expediency. Until engineering teams bring a greater degree of discipline to software component security, organizations will be exposed to a lot of hidden risk across their software portfolios.

We're not exaggerating here — our data found that 88% of Java apps scanned in the past year suffered from at least one component-based vulnerability.

JAVA APPLICATIONS WITH A VULNERABLE COMPONENT

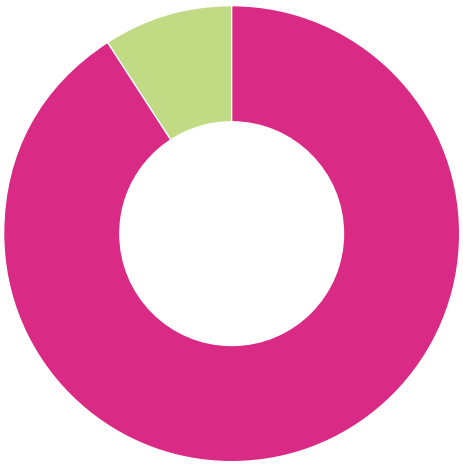


The trouble is that, in many cases, once a development team bakes open source components into their software, they rarely update them when security patches become available. So vulnerable components linger for far longer than they should, especially considering the way attackers jump on newly disclosed vulnerabilities to snag unpatched victims.

Take the highly popular Apache Struts package. This widely-used web application framework can be found across a wide range of applications in organizations small and large. Struts also has a history of a number of high-profile vulnerabilities, like the Struts-Shock vulnerability in Struts 2, which is at the root of breaches that have exposed tens of millions of records at financial institutions, hospitals, law firms, and government organizations.

JAVA APPLICATIONS USING STRUTS VERSIONS WITH A HIGH OR VERY HIGH SEVERITY VULNERABILITY

Percentage of Java applications using Struts



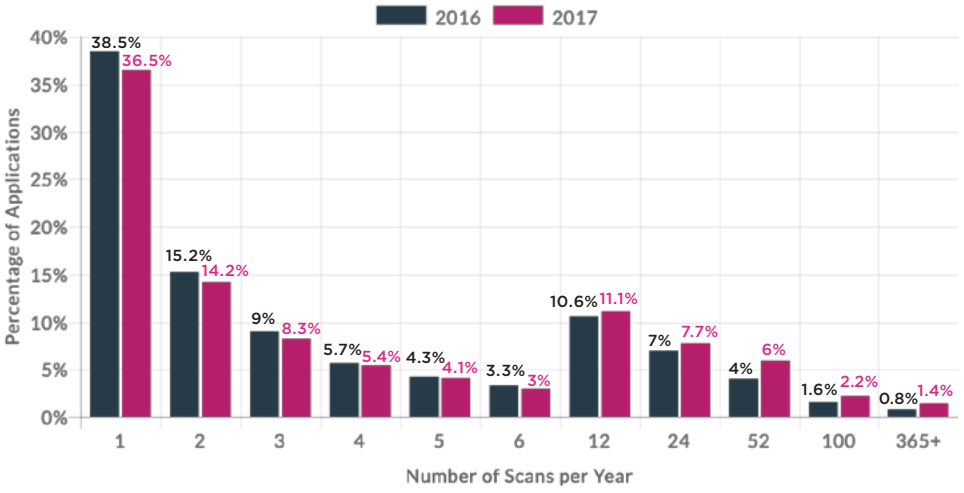
According to our scans between April 1 and September 30 of this year, 91% of all Java applications containing Struts components rely on a version of the framework that contains at least one high or very high severity vulnerability (a CVSS score of 6 or higher).

Finding ways to better monitor, track, and manage open source components in code is one of the highest impact activities developers can engage in today to move the needle on AppSec.

Level Up with DevSecOps

This is the second year that we've been following trends in our customer scanning behavior that connect DevOps practices with changes in application security testing. Our data suggests that the frequent and incremental changes spurred on by DevOps initiatives could also be increasing the rate at which organizations are scanning for security flaws.

SCAN FREQUENCY



As evidenced here, most applications in our portfolio last year were still scanned only once or a handful of times. But the subset of applications scanned monthly or more frequently is on the rise — more than 28% of applications fall into that bucket, a relative increase of 18% over the previous year. The biggest gains were in weekly scanning, which rose with a relative increase of 50% year-over-year.

SCAN FREQUENCY BY LANGUAGE

Language	Daily	At least weekly	At least monthly	At least quarterly	Less than quarterly
.NET	1%	8%	21%	14%	56%
Android	1.2%	2%	10.7%	11.2%	75%
ASP	0%	11%	10%	8%	71%
C++	0%	2%	12%	15%	70%
COBOL	0%	1%	7%	10%	83%
ColdFusion	0%	1%	13%	19%	67%
iOS	0%	2%	12%	10%	75%
iOS Swift	0%	2%	6%	8%	83%
Java	2%	12%	22%	13%	51%
JavaScript	0.4%	3%	9.5%	9.4%	77.7%
PHP	0%	3%	12%	9%	76%
Python	0%	0%	1.1%	7.5%	91.4%
VB6	1%	2%	5%	6%	74%

Digging deeper, we also looked at the data for scanning frequency by language.

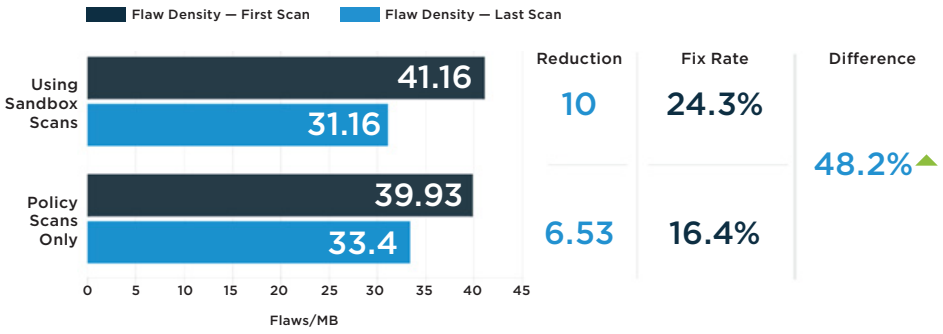
Low scanning frequency
Moderate scanning frequency
High scanning frequency

We found that the most prevalent languages on our platform, .NET and Java, were also the most frequently scanned languages. In Java, a healthy 35% of applications were scanned at least monthly. Now, this could legitimately point to adoption of DevOps practices within more mature organizations using an enterprise-friendly language. Other potential causational factors could be toolchain support, or maturity of CA Veracode support for that particular language.

One thing that is very solidly supported by our data is the fact that the increased velocity of scanning is helping organizations make a significant dent in their security flaws. The CA Veracode platform offers a feature for developers to scan their code without sending "official" policy results to their security and compliance minders. This sandbox mode allows developers to spot-check code throughout the development process. And when developers use sandbox scanning to frequently assess code along the way, their fix rate goes way up.

This year, applications scanned in a developer sandbox saw a 48.2% better fix rate than those with only official policy scans. It's yet another great proof-point that the incremental changes encouraged by DevOps have a huge upside for organizations that can streamline security testing throughout the SDLC.

SANDBOX SCANNING IMPACT ON FIX RATE



The takeaway here is that developers should be looking for every opportunity to integrate DevOps practices with security to reach for the goal of DevSecOps. An important part of DevSecOps is ensuring that the security toolchain is properly aligned with the DevOps toolchain. Highly automated DevOps organizations frequently fall down on security when their scanning tools don't tie directly into the integrated development environment (IDE) platforms that developers depend on. Integrations with developer environments are crucial for organizations hoping to build out their tooling to support DevSecOps.

Lean on Security Pros Like They Are Consultants

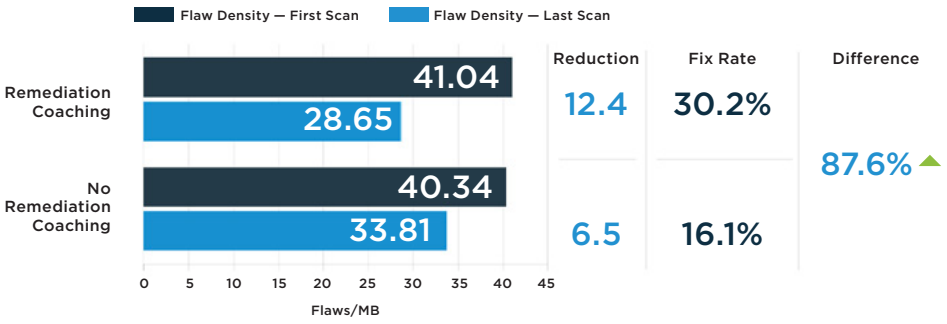
As DevOps practices proliferate and more organizations ask developers to do their own security testing, the role of the security team is changing. Long gone are the days when the security pros ran the scans, tossed a weighty document of vulnerability findings over the proverbial wall at the engineering team, and then ran for cover. As we mentioned, developers are increasingly tasked with running tests and fixing the code based on the findings. But that doesn't mean security people have been left twiddling their thumbs in the process.

When things are done right, security professionals are shifting into a consulting role, shoring up developers' security skills with advice on best practices, strategic planning, and one-on-one coaching.

If we're ever going to get to the point where we're bridging the gap between developers and the security team, there has to be a change in attitudes on both sides. Security teams will need to start catering to developers with a more service-oriented attitude of enablement. And developers must team with security staffers who offer domain expertise on things like those abuse case scenarios we discussed earlier.

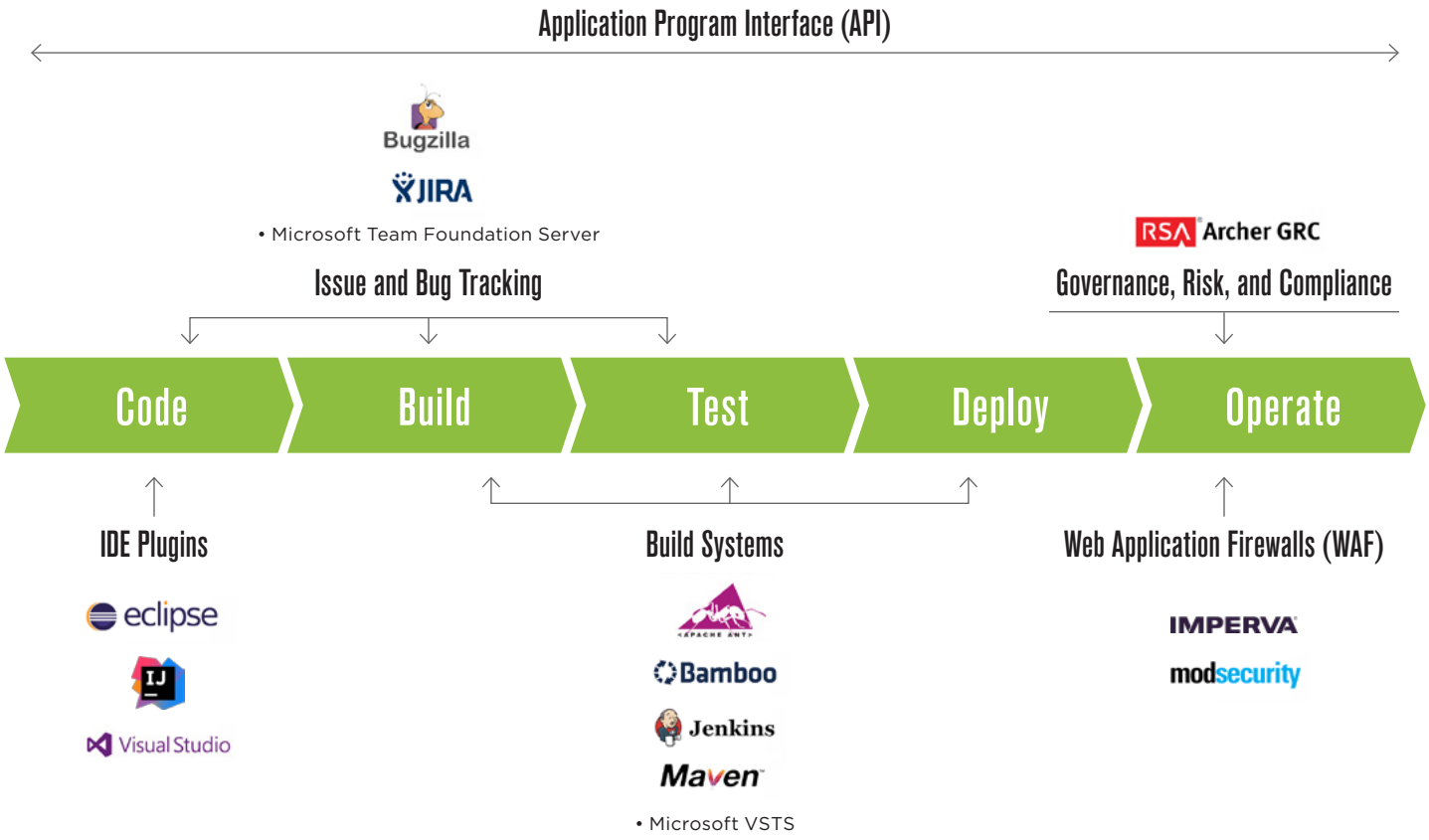
Developers who see security experts both internally and externally as a resource rather than as an adversary tend to make big gains on risk reduction within application portfolios. We saw an example of this in the data for this year's report. Developers who paired their vulnerability findings with remediation coaching from security experts saw an impressive 87.6% better fix rate than those who received no coaching.

REMEDIATION COACHING IMPACT ON FIX RATE



At the end of the day, security experts can help orient developer thinking, guide future learning, and answer timely questions. On a more strategic level, though, developers can work with security teams to acquire and integrate tools into the developers' toolchain in such a way that security doesn't pose a huge speed bump in their development process. Ideally, the whole point of DevSecOps is to both increase velocity of the SDLC, while ensuring that security testing and remediation is conducted at every stage of the lifecycle.

SECURITY TESTING INTEGRATIONS FOR DEVSECOPS



Developers would do well to work with the security team to ensure that the security testing mechanisms they have in place will not only integrate with security systems like web application firewalls and GRC systems, but also with the tools that developers use to get work done every day. That includes IDE systems, ticketing and bug tracking, and build and automated deployment systems that are the lifeblood of any DevOps or Agile toolchain today. We understand that this will take a huge commitment from the security team. But this is a two-way street, and developers will also need to collaborate with security to test faster and smarter, and use the results of their tests to ship better code in the long run.

Fight for Training and Use It

The majority of IT pros, security pros, and developers agree that developers today don't get enough training to help them support AppSec efforts in the enterprise, according to the 2017 DevSecOps Global Skills Survey conducted by Devops.com.

This is a huge missed opportunity. Our data shows that training can make a measurable improvement on fix rates, particularly when paired directly with information about existing vulnerabilities in the code base.

DEVELOPERS DON'T GET THE TRAINING THEY NEED

86%

of IT pros say their orgs don't spend enough on AppSec training.

76%

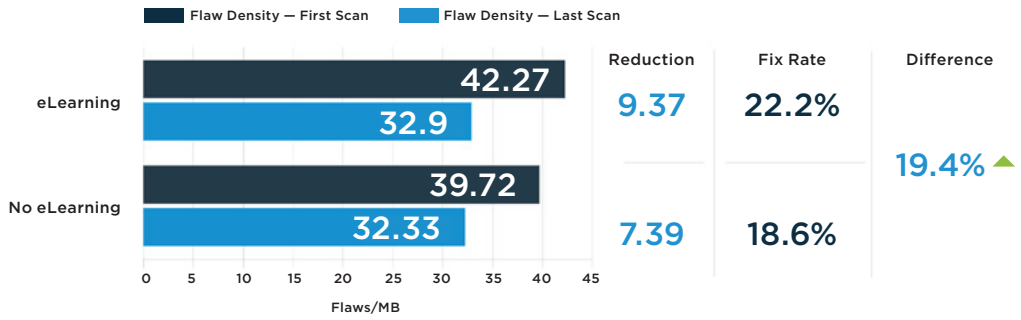
say they weren't required to take a course in security while in college.

68%

say their organizations don't provide adequate AppSec training.

Just the act of going through simple eLearning modules about common vulnerabilities can help a developer improve their fix rate by an average 19.4% above those who don't have access to eLearning.

ELEARNING IMPACT ON FIX RATE



Developers without the resources needed to improve AppSec knowledge must find a way to fight for training if it isn't available. We've already talked about how security teams can be a good partner for developers. One consideration is teaming up with security colleagues to make the case to the higher-ups. This two-pronged approach could make a difference when making the business case for increased training budgets.

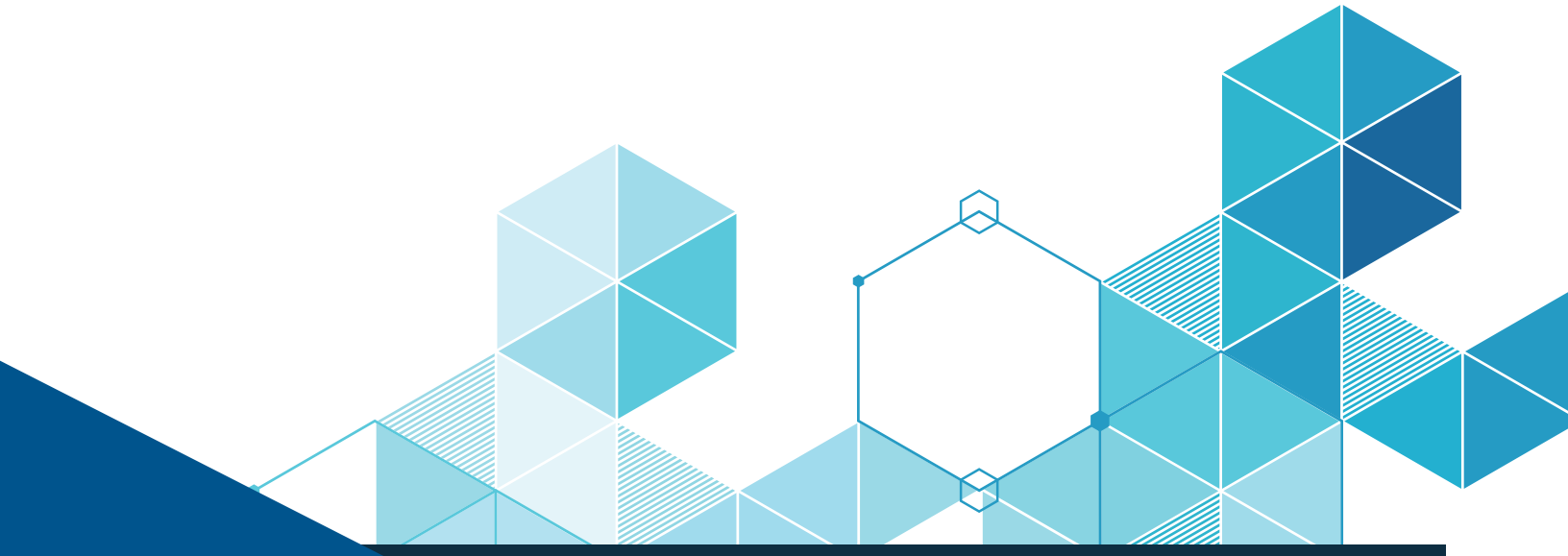
Conclusion

AppSec Depends on You

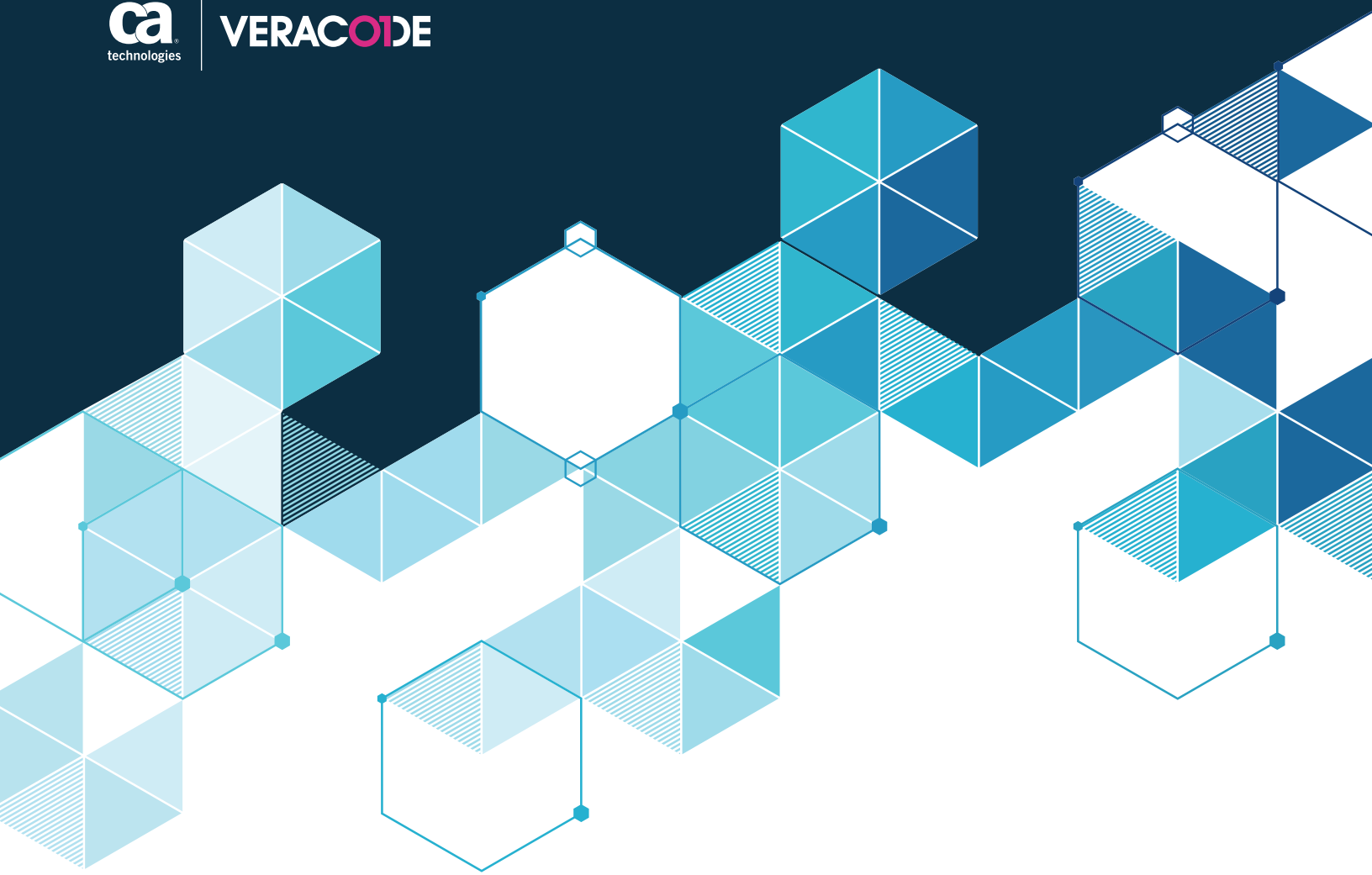
As we've seen in the data for this year's State of Software Security report, developers are essential to finding and fixing vulnerabilities that plague the majority of applications. It's therefore not surprising that hiring managers and IT leaders are seeking out development pros with security knowledge. Developers have a lot more responsibility for security than ever before, as processes shift to DevOps and, increasingly, DevSecOps.

In the near future, developers will need to fit the profile of what we call the full spectrum engineer (FSE). An FSE is someone who is not just good at keeping up with the latest trends in design and implementation, but knows how to test for quality, performance, and security. The most desirable developers understand the complexities of deployment, and know how important it is to ensure code looks and behaves the same whether running in development, QA, staging, or production. FSEs are software engineers who take part in pager duty with the rest of the team and build software that can self-detect and even self-heal when not running properly.

The biggest trait of these developers of the near future is accountability — and a huge part of that is ensuring that security is considered from the beginning of the process. Ultimately, security is just as much a part of acceptance criteria as any other non-functional requirement. We hope that the recommendations in this report will help you better understand your role in application security and get to the next level of competency to master all the requirements and expectations of making great software. It's a lot of work, but following up on these key recommendations is not only good for your organization. It's also good for developers and your career.



ca technologies | VERACODE



STATE OF SOFTWARE SECURITY 2017

Read the Full Report
veracode.com/soss

Contact Us
To Learn More or Request a Demo

ABOUT CA VERACODE

Veracode, CA Technologies' application security business, is a leader in helping organizations secure the software that powers their world. Veracode's SaaS platform and integrated solutions help security teams and software developers find and fix security-related defects at all points in the software development lifecycle, before they can be exploited by hackers. Our complete set of offerings help customers reduce the risk of data breaches, increase the speed of secure software delivery, meet compliance requirements, and cost effectively secure their software assets – whether that's software they make, buy or sell. Veracode serves over a thousand customers across a wide range of industries, including nearly one-third of the Fortune 100, three of the top four U.S. commercial banks and more than 20 of the Forbes 100 Most Valuable Brands. Learn more at veracode.com, on the Veracode Blog, and on Twitter.