2017 State of Application Security: Balancing Speed and Risk

A SANS Survey

Written by Jim Bird
Advisors: Eric Johnson, Barbara Filkins and Frank Kim

October 2017

Sponsored by
Rapid7, Synopsys, Tenable, Veracode, and WhiteHat Security
The speed of software development is accelerating—and so are software security risks. Large software development projects that used to take years to complete have been outpaced by smaller, agile teams that deliver working software every few weeks. High-speed cross-functional DevOps teams are pushing software changes directly to production, sometimes hundreds or even thousands of times each day. Organizations are taking advantage of cloud platforms and on-demand services, containerization, and automated build and continuous delivery pipelines to accelerate delivery cycle times and cut costs to the bone.

All of this radically changes how development teams—and their security/risk management teams—think and work.

What does security look like in a world of continuous change? How can security teams possibly keep up, since they rely only on gate reviews and penetration testing to understand and control risk? What security procedures, tools and practices work better in a high-velocity development program? And, can agility and velocity be used to improve security?

In our fifth annual survey on application security,1 214 IT professionals responded to these questions. We wanted to learn how respondents are balancing speed and risk, so we compared the results of fast development teams that push out new programs and updates in a week or less to the results of slow teams, which take longer.

We compared how respondents test applications being pushed out into production, including what tools respondents’ organizations used, how often and when they tested their applications, who was responsible for testing, and how satisfied they were with their application security (AppSec) programs overall.

1 Check out the previous application security surveys:
We also looked at how quickly and effectively they fixed problems. What we found was that application security assessment is, on the whole, moving faster. But some organizations are falling far behind in their testing: 24% rely on testing security once a year or less, much too infrequently to support the increased speed of development, while 10% still are not testing or assessing their business-critical applications at all. Most organizations are still relying heavily on audits and external reviews, pen testing and other manually intensive processes to find security vulnerabilities.

The good news is that organizations able to make changes to their code more quickly are also fixing more security vulnerabilities than their slower-moving competitors. They are achieving this by breaking down organizational silos, moving more responsibility for security testing directly to developers or cross-functional teams—and by taking advantage of end-to-end workflow automation, which integrates security into Agile and DevOps toolchains so they can test security faster and more often.

These and other risks and best practices are reported in the remainder of this paper.
Application security risks and threats are constantly changing. In this survey, more than 15% of organizations experienced breaches related to their applications in the past two years. While the major contributors to security incidents continued to be public-facing web applications and Windows OS, followed by legacy applications, we saw an increase in successful attacks against applications in the cloud—and now against containers.

Survey Background

Respondents came from a wide range of industries, including banking or finance (18%), technology (17%), cyber security services (10%), healthcare (8%) and application development firms (8%). Most respondents were from the United States (72%), with global representation across all sizes of organizations from small (up to 1,000 employees, 37%) to very large (over 50,000 employees, 20%). Reflecting the SANS community, 69% of respondents worked in security- or compliance-related roles, from hands-on administrators and analysts to senior managers and C-level executives.

For much of this survey, we sorted answers based on how frequently respondents’ organizations deployed changes to their production systems:

- **Fast (and really fast)**—deploying changes weekly, daily or continuously (several times per day), tending to follow more agile and lean incremental change approaches, including DevOps and continuous delivery
- **Slow**—rolling out changes monthly, quarterly or annually, following a more traditional approach to change

Let’s look at where organizations face risk, how they address risks, what tools and practices they rely on, and what their priorities are.
Risk at the Application Level

Organizations continue to be mainly focused on protecting public-facing web applications and other custom applications developed in-house. Applications in the cloud (private clouds and to a lesser extent public clouds) and mobile apps are also important areas of focus, as illustrated in Figure 1.

What types of applications are you protecting under your AppSec program? Select those that most apply.

**Figure 1. Protecting Application Portfolios**

APIs are becoming a specific area of focus for 42% of organizations, and 28% of organizations are now dealing with applications hosted in containers such as Docker. In our 2016 survey, we asked which apps organizations were spending their resources on, and the answers were similar: public-facing web apps, followed by legacy apps, then customized apps, mobile apps and APIs.\(^2\)

---

Risks and Breaches

Over the past two years, 15% of organizations responding to this survey experienced a breach, and, alarmingly, 21% don’t know whether they experienced a breach where applications were the source. This number is lower than in our 2016 survey, in which 23% of respondents reported their applications were the source of their breaches.3

This year, the biggest sources of breaches continued to be public-facing web applications and Windows OS, closely followed by legacy applications (which are often left untested because security teams either aren’t aware of them or don’t have access to their source code). Custom applications are another common target of attack. We are also seeing more successful attacks against APIs and applications in the cloud—and now containers, as shown in Figure 2.

TAKEAWAY:
While known vulnerabilities are routinely being exploited, real risks and threats are continuously changing as the attack surface of each organization increases. Security programs need to keep up with changes to the threat landscape and adapt, even as they struggle to successfully implement foundational practices such as the CIS Controls.4

---


4 www.cisecurity.org/controls
**Speed Versus Breaches**

In looking at respondents that experienced a breach and comparing their breach experience based on their speed of deploying changes, organizations that are changing continuously, daily or weekly are not experiencing more problems than organizations that make changes only annually. See Figure 3.

**Over the past two years, have any of your applications been the source of breaches, attacks on others or sensitive data leaks?**

![Figure 3. Breaches Compared to Frequency of Change](image)

**Risk at the Language Level: New Languages, New Risks**

Because different programming languages and toolsets present different challenges and opportunities to engineering and security teams—directly affecting how they deliver and test—it is important to understand security risk at the language and library level. See Figure 4.
Java and .NET continue to be major sources of security risk because they are still the most commonly used enterprise application development languages. However, JavaScript has recently overtaken .NET as a risk concern, reflecting its increasing popularity as a lighter-weight alternative. In 2016, Java led as the source of risk for 55% of respondents, followed by .NET for 44% and JavaScript for 40% of respondents.\(^5\)

JavaScript is widely used to develop client applications, taking advantage of powerful front-end frameworks, such as Angular(JS), React and Ember (and libraries such as JQuery), and increasingly for server-based applications using Node.JS. These frameworks are an additional source of security risks. JavaScript and other dynamic scripting languages, for example PHP and Python, are also more difficult to check at build time than static languages, which means that more problems can escape to be found at runtime.

C/C++ continues to be a source of risk both because of lack of safe programming constructs and because these languages are often used to solve low-level programming problems, such as OS and platform services, device drivers or real-time/embedded software.

---

**Polyglot Programming: Flexibility Brings New Risks**

Polyglot programming, where development teams write code simultaneously in several different languages, is increasingly common in modern Agile and DevOps (continuous delivery/continuous integration) environments.

In polyglot programming, developers are encouraged to choose different languages, frameworks and runtimes based on what they believe is best suited to the specific problem they may be trying to solve or to learn about a new language or tool set. In microservices environments, where small, self-directing teams are each responsible for a specific service, polyglot programming can result in hundreds of different technologies that need to be tracked, understood and secured.

Automated toolchain support and even integrated development environment (IDE) support may be limited or nonexistent for new languages and frameworks. This is especially true for Static Application Security Testing (SAST) and software component analysis (SCA) tools, both of which constitute an important part of many security assessment programs, as we’ll see in this analysis. Organizations will need to develop secure coding guidelines, as well as review and assess their application frameworks for security capabilities and risks.

---

Platform Risks: Cloud and Containers

Cloud platforms and, more recently, containers are becoming an important part of IT programs to reduce operational costs and increase agility. Organizations can take advantage of scale, standardization and on-demand capacity for what Netflix, a pioneer in this space, calls “undifferentiated heavy lifting” and “NoOps”: simplifying and abstracting operations and making it transparent to developers.

Cloud services and containers allow developers to provision their own infrastructure on the fly, making it even easier and faster for them launch new applications—and to make mistakes that could have an impact on reliability and security. However, they also introduce risks around identity and access control, untrusted images, security orchestration, container “breakouts” and more.

As you can see from Table 1, 21% are currently hosting apps in the public cloud, and 31% plan to have apps running in the public cloud within next 2 years.

<table>
<thead>
<tr>
<th>Where are applications hosted?</th>
<th>2017</th>
<th>Next 2 Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public cloud</td>
<td>21.4%</td>
<td>30.5%</td>
</tr>
<tr>
<td>Private cloud</td>
<td>27.8%</td>
<td>31.9%</td>
</tr>
<tr>
<td>Hybrid</td>
<td>11.3%</td>
<td>15.6%</td>
</tr>
<tr>
<td>On-premises/Traditional data center</td>
<td>63.3%</td>
<td>49.4%</td>
</tr>
</tbody>
</table>

Security teams must catch up and understand these architectures and how to keep them secure.
Organizations continue to depend heavily on monitoring (IDS), vulnerability scanning, and identity and access management (IAM)—all classic security controls. Security training for developers is also seen as key, although not as important as in our 2016 survey, where it was by far the most valuable practice. Least useful: the sexy new stuff (such as Runtime Application Self-Protection [RASP] and cloud-based controls) and virtual patching, which, as we saw in earlier surveys, requires a high level of coordination between development, operations and tool suppliers. In Figure 5, we look at which security practices are used by slow- and fast-moving organizations.

Figure 5. Comparing Speed of Change and Security Practices Used


7 See the series of AppSec surveys:


Continuous monitoring and continuous vulnerability scanning are especially important in fast-moving organizations that deploy changes at least weekly to catch problems that might get past testing and reviews. Security architecture and design reviews are not used as much as they are in slower-moving organizations, where reviews can be added as a stage gate. In fast-moving, iterative development environments, it is not obvious whether and when reviews should be scheduled and how they should be done.

Managing Cloud Risks

The same controls used to protect traditional data centers are also the most common controls used to protect systems in the cloud: Most organizations are relying on classic network security protection (IDS/IPS, firewalls) and account management controls, as shown in Figure 6.

What runtime cloud protection solutions do you use? Select all that apply?

![Figure 6. Protecting the Cloud](image)

Only a handful of organizations are taking advantage of newer cloud-based runtime protection services, including microsegmentation, cloud access security brokers (CASBs) or RASP. However, a significant percentage is using encryption, auditing and other mechanisms to protect data and customer privacy in the cloud.
Moving to the Cloud for Security Reasons

The arguments in favor of the cloud for operational cost savings are obvious, especially to online startups and businesses with highly-varied demand cycles. But many organizations—especially enterprises and government agencies—have resisted moving to cloud services because of security, privacy and compliance reasons.

This is now changing, as major cloud providers continue to make massive investments in infrastructure security and availability, expanding and improving their operational controls, and now offering comprehensive security and compliance capabilities as part of their platforms.

Today, organizations are moving to the cloud not only because of operational economies of scale, but also to take advantage of these security and compliance strengths. One example is Capital One, whose CIO has gone so far as to state that he believes that, by leveraging the compliance and security services of its key cloud service providers, its applications are safer in the public cloud than in its own data centers.8

8 https://aws.amazon.com/solutions/case-studies/capital-one
Keeping Up with the Rate of Change

Although 10% of respondents say they aren’t doing any security testing at all, 85% of respondent organizations are assessing or testing the security of their mission-critical applications. Of these, 12% are doing security testing on a continuous basis. At the other extreme, 24% of all organizations are still relying on testing once a year or less.

Fast-moving organizations are more likely to be following Agile or DevOps practices such as continuous delivery.

Fast-moving organizations (those deploying continuously, daily or weekly) made up 43% of the survey base—of those, only a small percentage are pushing changes continuously (5%). The remainder (57%) deployed more slowly.

Security Testing and Speed of Delivery

Teams that are moving faster should also be testing faster. To understand whether security testing is keeping up with the speed of delivery, we compared how often organizations make changes to how often they do security assessments. Results from organizations considered to be “fast” are highlighted in green. See Table 2.

Table 2. Frequency of Security Assessment/Testing Compared to Change Rate

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Changes</th>
<th>Assess/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuously (several times each day)</td>
<td>5.3%</td>
<td>11.9%</td>
</tr>
<tr>
<td>Daily</td>
<td>12.0%</td>
<td>11.3%</td>
</tr>
<tr>
<td>Weekly</td>
<td>25.4%</td>
<td>19.1%</td>
</tr>
<tr>
<td>More than once per month</td>
<td>17.7%</td>
<td>12.5%</td>
</tr>
<tr>
<td>Monthly</td>
<td>18.7%</td>
<td>19.1%</td>
</tr>
<tr>
<td>Quarterly</td>
<td>13.4%</td>
<td>17.3%</td>
</tr>
<tr>
<td>More than once per year</td>
<td>3.8%</td>
<td>13.1%</td>
</tr>
<tr>
<td>Annually</td>
<td>1.9%</td>
<td>21.4%</td>
</tr>
<tr>
<td>Less than once per year</td>
<td>1.9%</td>
<td>2.4%</td>
</tr>
</tbody>
</table>

Looking at the entire sample, that appears to be the case. But, do teams that move faster also test more often? Table 3 reveals that the faster the development environment, the more frequently testing is done.

Table 3. Frequency of Testing Based on Fast or Slow Rate of System Change

<table>
<thead>
<tr>
<th>Speed of System Change</th>
<th>Continuous</th>
<th>Daily</th>
<th>Weekly</th>
<th>More than Once/Month</th>
<th>Monthly</th>
<th>Quarterly</th>
<th>More than Once/Year</th>
<th>Annually</th>
<th>Less</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>8.54%</td>
<td>7.93%</td>
<td>9.76%</td>
<td>6.71%</td>
<td>7.32%</td>
<td>8.54%</td>
<td>3.66%</td>
<td>8.54%</td>
<td>0.61%</td>
<td>1.22%</td>
</tr>
<tr>
<td>Slow</td>
<td>3.66%</td>
<td>3.66%</td>
<td>9.15%</td>
<td>5.49%</td>
<td>12.20%</td>
<td>7.32%</td>
<td>8.54%</td>
<td>13.41%</td>
<td>0.61%</td>
<td>1.83%</td>
</tr>
</tbody>
</table>
What Is Driving the Need for Speed?

Time to market drives speed, of course, as organizations outpace competitors to deliver a new idea or service and establish market leadership. But there are many more reasons to speed up software delivery and the related security controls, such as:

- Shaping design using feedback from users in production through A/B experiments and controlled feedback loops
- Putting changes into the hands of users early to see what they like or don’t like and what they use and don’t use, instead of guessing and missing—or overdesigning
- Reducing development and delivery costs by eliminating waste, automating and standardizing work, and reusing code components
- Reducing project risks and business risks by quickly delivering minimum viable products (MVPs) stripped to the essentials, so that organizations can learn whether they are on the right track, and pivot toward a new design or use, or fail early and save resources
- Reducing operational risks by breaking big projects into smaller and simpler changes that can be tested and delivered in steps, eliminating the risks and impact of big bang rollouts

Leveraging speed in these ways has led to the incredible success of organizations such as Amazon and Google, enabling them to achieve true agility, or continuous delivery, at scale.

Testing and Delivery Velocity

As engineering teams continue to accelerate delivery, security personnel need to speed up security assessments. Security teams have traditionally depended on manual gate reviews in waterfall projects, especially audits and pen testing—practices that are mandated by compliance regimes such as PCI DSS. But how do you fit gate reviews and pen testing into continuous iterative development and delivery?

Even automated scanning can take hours or days to complete for large applications. That won’t work for apps that are deployed daily or several times per day.
Risks and Opportunities

To keep up with high-velocity delivery teams, security testing needs to be automated, fast, incremental, and made an in-line part of development and delivery workflows and pipelines.

Increased speed in Agile, DevOps and continuous deployment introduces new risks, including:

- Changes being made so quickly, and so often, that it is difficult to understand and review them for risk
- Lack of stage gates in iterative, incremental development and continuous flow, which means there are no natural points to insert reviews, tests or other controls
- Not enough time to do exhaustive testing or reviews before changes get pushed to production
- Constantly changing design, which means that the risk profile is also constantly changing

Speed introduces new opportunities to reduce risk, too:

- Frequent delivery drives teams to automate and standardize workflows, especially build-and-deploy pipelines, increasing control over and transparency into change, and reducing risk of unauthorized changes or insider attacks.
- Most changes are incremental and small, which makes it easier to understand and test, and safer to release each change.
- Research shows that constantly changing the attack surface of a system can make an attacker’s job more difficult.9

---

9 https://pdfs.semanticscholar.org/1148/f37a8ca0a5ca0a26178c7d85a063bd539725.pdf
Security Testing Tools and Practices

As illustrated in Figure 7, most organizations are still heavily dependent on manual testing and reviews, including pen testing and external compliance audits (required by regulations such as PCI DSS). This reflects the importance of compliance in driving security programs and controls, something we have looked at in earlier application security surveys.

How does your organization test applications for vulnerabilities? Select all that apply.

- Internal penetration testing
- Third-party penetration testing
- Automated code review or Static Application Security Testing (SAST)
- Compliance reviews or audits by a third party
- Manual code review
- Dynamic Application Security Testing (DAST)
- Container security scanning
- Open source and third-party component analysis
- Interactive Applications Security Testing (IAST)
- Runtime Application Self-Protection (RASP)
- Other

Figure 7. Testing for Vulnerabilities

Following pen testing, respondents selected automated code reviews (SAST), compliance audits, manual code reviews and testing by automated application scanning (DAST). In addition, 28% of organizations are scanning containers for security vulnerabilities, and 26% are using open source and third-party component analysis.

TAKEAWAY:
Pen testing and third-party reviews are still key parts of security programs, even for organizations that are pushing out changes several times a day.
Automating Continuous Testing

To move fast, developers need to rely heavily on automated testing that fits into Continuous Integration and Continuous Delivery (CI/CD) cycles. Automated unit testing, which is the backbone of functional testing for Agile development teams, is good for finding regressions, but poor at finding vulnerabilities. Teams need to find other tools and approaches that support rapid cycling, such as:

- SAST/automated code review tools integrated into automated builds or directly into developer's IDEs to catch mistakes as developers make changes
- Manual code reviews done as part of the code check-in workflow, using code-review management tools to help developers request and respond to reviews and track the results
- Automated software component analysis (SCA) for open source and third-party libraries, integrated into automated builds, and as part of code check-in
- Container vulnerability and security scanning integrated in similar ways
- DAST application scanning run as part of automated functional testing and acceptance testing

Trade-off with Automation

The faster teams move, and the more they rely on automation, the more tradeoffs they need to make. Because not enough time is available to run deep, exhaustive scans or other security tests in continuous testing, organizations need to scan first for the most critical vulnerabilities. Then they need to target recently changed code for incremental testing and rely on smoke tests to catch other critical mistakes. Rules and tests that take too long to run or are too noisy need to be tuned or cut out, leaving holes in test coverage.

This means that periodic pen testing, in-depth manual reviews, configuration auditing, deep scanning and fuzzing are still needed to find errors that escape tight automated loops.
**Developer Testing on the Rise**
Over the past three years, the number of organizations relying on development teams to do security testing has increased from 22% (2015) to 30% (2016), and now to 51% (2017).

**Security Testing Personnel**

External parties (auditors, pen testers, scanning services) and internal security teams are primarily responsible for security testing and assessments, while development teams and system architects are primarily responsible for corrective actions, according to respondents. See Figure 8.

However, an increasing amount of responsibility is being assigned to cross-functional teams (across dev/ops/sec), and directly to developers—especially in faster organizations.

---

**Who is responsible for running the application security testing for your organization or work group? Who is responsible for final acceptance of the testing results and any corrective actions resulting from that testing? Select all that apply to your organization.**

![Responsibility for Testing](image-url)

*Figure 8. Responsibility for Testing*
Vulnerabilities Discovered

Most organizations (60%) find between one and 25 vulnerabilities per month. A small percentage finds more than a thousand per month. See Table 4. But the majority of the problems that are being found are not critical, as shown in the Table 5.

<table>
<thead>
<tr>
<th>Number of Vulnerabilities Found per Month</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>6.8%</td>
</tr>
<tr>
<td>1–25</td>
<td>60.3%</td>
</tr>
<tr>
<td>26–50</td>
<td>11.8%</td>
</tr>
<tr>
<td>51–100</td>
<td>9.3%</td>
</tr>
<tr>
<td>101–250</td>
<td>4.4%</td>
</tr>
<tr>
<td>251–500</td>
<td>1.9%</td>
</tr>
<tr>
<td>501–1,000</td>
<td>1.2%</td>
</tr>
<tr>
<td>&gt;1,000</td>
<td>4.4%</td>
</tr>
</tbody>
</table>

This indicates that teams are wasting time (sometimes a lot of time) dealing with false positives, low fidelity findings and other noise in security testing.

Looking at testing results through a velocity lens shows that moving too fast can create risks when it comes to security testing, a relationship made clear in Figure 9.

Teams with the most rapid development procedures are also finding fewer vulnerabilities. Earlier results indicated that fast-moving organizations are also doing more frequent scanning and testing. This may indicate that they are doing a more superficial job of security assessment, because they need to fit their testing into fast feedback cycles—as we've explained, security testing takes time to do right.
Vulnerabilities Repaired

Faster organizations are more likely to fix vulnerabilities than their slower competitors, because the costs and risks of change in faster organizations are generally lower: The more often you do something, the better you get at it. See Figure 10.

**What percentage of critical security vulnerabilities does your organization repair satisfactorily and in a timely manner?**

*Figure 10. Comparing Rate of Change to Speed of Vulnerability Repair*
Overall, respondents reported that 41% of serious or critical vulnerabilities are fixed within a week of when they were found. They fix an additional 34% of their vulnerabilities within one month. See Figure 11.

On average, how long does it take for your organization to fix and deploy a patch to a critical application security vulnerability for systems already in use?

![Image of bar chart showing time needed to deploy a patch.

In fact, looking back year over year, all organizations are getting faster at fixing vulnerabilities in production, based on PCI’s 30-day patch rule. In 2016, 66% of respondents’ organizations achieved such levels of success, improving to 75% in 2017. See Table 6.

Table 6. Time to Patch a Vulnerability 2016–2017

<table>
<thead>
<tr>
<th>Time to Correct a Vulnerability</th>
<th>2016</th>
<th>2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same day</td>
<td>6.0%</td>
<td>5.9%</td>
</tr>
<tr>
<td>Next day</td>
<td>7.5%</td>
<td>5.9%</td>
</tr>
<tr>
<td>2–7 days</td>
<td>26.0%</td>
<td>29.6%</td>
</tr>
<tr>
<td>8–30 days</td>
<td>26.0%</td>
<td>33.7%</td>
</tr>
<tr>
<td>31–90 days</td>
<td>14.9%</td>
<td>6.5%</td>
</tr>
<tr>
<td>91–180 days</td>
<td>6.4%</td>
<td>4.7%</td>
</tr>
<tr>
<td>6 months to 1 year</td>
<td>3.2%</td>
<td>1.2%</td>
</tr>
<tr>
<td>More than a year</td>
<td>0.7%</td>
<td>0.6%</td>
</tr>
<tr>
<td>Unknown</td>
<td>8.5%</td>
<td>8.9%</td>
</tr>
</tbody>
</table>

Organizations are completing most (53%) vulnerability repairs through patches or upgrades to the runtime environment, 47% are handled at root cause by secure software development life cycle (SDLC) practices, and another 47% are completed by patching third-party or open source components, as shown in Figure 12. Automated testing and deployment, for example in Continuous Delivery, make this easier and safer to do.

Note that the 47% of vulnerabilities that are corrected using root cause analysis identify problems in the SDLC. Understanding and fixing problems at the root cause takes time, for both agile and traditional organizations. This may be easier in Agile and DevOps environments because they encourage frequent and blameless retrospection and introspection, as well as continuous improvement. Agile teams should be more attuned to identifying root causes and acting on remediation plans.
Although much attention and money are invested in technology for secure development and testing, the biggest challenges organizations face in their application security programs involve people, not tools. See Figure 13.

**What are your top three challenges in implementing application security for production systems at your organization?**

*Indicate the top three in no particular order.*

- Bridging the gap between software development, security and compliance
- Silos between security, development and business units
- Lack of funding or management buy-in
- Lack of application security skills, tools and methods
- Shortage of technical resources to maintain security in production applications
- Lack of integrated security and remediation life-cycle workflow
- Fear of modifying production code (might “break the app”)
- Identifying all applications in the portfolio
- No clear definition of success (metrics, CSFs)
- Poor remediation, workflow and advice for fixing discovered vulnerabilities
- Waiting for service releases to fix problems
- Testing applications containing no source code (e.g., commercial off-the-shelf apps, third-party components)
- Lack of testing support for applications written in legacy languages
- Developing test scenarios or test cases that address security
- Visibility into containers
- Lack of testing support for applications containing new frameworks
- Other

**Figure 13. AppSec Challenges**
Bridging cultural and communications gaps and organizational silos, obtaining management buy-in, and dealing with a lack of security skills are all management problems. But respondents’ organizations are finding ways to overcome some of these problems. See Figure 14.

More than 45% attribute their ability to overcome challenges to adopting more effective testing methods across the SDLC, building cross-functional teams, and encouraging communications across teams and silos. More integrated technology is also playing an important role in breaking down silos: end-to-end testing, automated end-to-end workflows and integrated tools are helping to bridge gaps between teams and reduce risks.
People and process must come first. Technology is simply helping facilitate the workflow of a DevOps culture. These are difficult and deep organizational, cultural and management problems, which aren't under the control of the security team or engineering teams to solve. Organizations are finding ways to solve these problems by building bridges between engineering and security teams using the following tools:

- Full life-cycle testing, all the way to deployment
- Cross-functional teams across dev/ops/sec
- Communications plans across teams
- Automated end-to-end workflows
- Integrated testing and development tools

These practices are all encompassed under what is being called DevOpsSec or DevSecOps: a collaborative, open approach to integrating engineering, security and compliance teams.

**DevOps and DevSecOps/DevOpsSec**

Breaking down silos, creating cross-functional teams, automating end-to-end workflows and testing, open communications, and transparency are all hallmarks of what is being called DevSecOps or DevOpsSec today.

DevOps is about applying Lean and Agile development principles, values, practices and end-to-end automated workflows to the release, deployment and operations of systems. Key ideas in DevOps include the following:

- Small, self-directing, highly collaborative cross-functional teams across development and operations
- Small, incremental (continuous) improvements
- Teams that are responsible and accountable for development, deployment, operations and support for the life of the system or service (you build it, you run it)
- Continuous delivery and deployment, where automated and repeatable build and deployment pipelines promote changes from development through testing, staging and production
- Infrastructure as code, which defines infrastructure configuration in code, and makes changes to configuration using the same type of automated pipelines as application changes

DevSecOps, or DevOpsSec, or sometimes Rugged DevOps, brings security (and where possible compliance, known as continuous compliance in DevOps circles) into the same model to create collaborative, open, transparent teams of people working across development, operations and security to understand and solve security problems together.
DevOps is moving faster, and security teams can leverage the speed of change to get security patches out more quickly and cheaply, closing the window of vulnerability to attack. Speed becomes an important security advantage.

As organizations continue to speed up, they are fundamentally changing how people think and work. Instead of big, long-running software development projects with waterfall handoffs between silos and outsourced maintenance, more work is being delegated to small, self-directed engineering teams responsible for building and operating services (or microservices and containers), taking pressure off operations/release management and security teams, and also eliminating bottlenecks in the workflow.

Faster decision making and faster delivery mean that security specialists need to get closer to engineering, so that engineers and security personnel can work together to identify and understand risks and manage them on a continuous basis.

This calls for people, process and technology changes that will bring dev/ops/sec together in cross-functional teams, providing more touchpoints, visibility, standardized workflows and transparency. However, this creates a critical scaling problem for AppSec programs in which skills are already in high demand. You will also need to push more responsibility for security directly to development and engineering teams, giving them training so that they understand more about AppSec risks and how to deal with them, and finding them automated tools that fit into how they actually think and work: iteratively, incrementally and rapidly. In that sense, security teams become enablers and coaches instead of enforcers and blockers.
Jim Bird, SANS analyst and co-author of DEV534 Secure DevOps, is an active contributor to the Open Web Application Security Project (OWASP) and a popular blogger on agile development, DevOps and software security at his blog, “Building Real Software.” He is the CTO of a major U.S.-based institutional trading service, where he is responsible for managing the company’s technology organization and information security program. Jim is an experienced software development professional and IT manager, having worked on high-integrity and high-reliability systems at stock exchanges and banks in more than 30 countries. He holds PMP, PMI-ACP, CSM, SCPM and ITIL certifications.

Barbara Filkins, a senior SANS analyst, holds several SANS certifications, including the GSEC, GCIH, GCPM, GLEG and GICSP, the CISSP, and an MS in information security management from the SANS Technology Institute. She has done extensive work in system procurement, vendor selection and vendor negotiations as a systems engineering and infrastructure design consultant. Barbara focuses on issues related to automation—privacy, identity theft and exposure to fraud, as well as the legal aspects of enforcing information security in today’s mobile and cloud environments, particularly in the health and human services industry, with clients ranging from federal agencies to municipalities and commercial businesses.

Eric Johnson, the Application Security Curriculum product manager at SANS, is the lead author and instructor for DEV544 Secure Coding in .NET, as well as an instructor for DEV541 Secure Coding in Java/JEE. A senior security consultant at Cypress Data Defense, Eric’s experience includes web and mobile application penetration testing, secure code review, risk assessment, static source code analysis, security research and developing security tools. He currently holds the CISSP, GWAPT, GSSP-.NET and GSSP-Java certifications.

Frank Kim leads the management and software security curricula for SANS, developing courses on strategic planning, leadership and application security. He is also a SANS certified instructor, helping to shape, develop and support the next generation of security leaders. Previously, Frank served as CISO at the SANS Institute, leading its information risk function, and executive director of cybersecurity at Kaiser Permanente, where he built an innovative security program to serve one of the nation’s largest not-for-profit health plans and integrated healthcare provider. Currently, as founder of ThinkSec, a security consulting and CISO advisory firm, Frank helps leaders develop business-driven security programs.
SANS would like to thank this survey’s sponsors:

- Rapid7
- Synopsys
- Tenable
- Veracode
- WhiteHat Security
Upcoming SANS App Sec Training