

STATE OF SOFTWARE SECURITY

VOLUME 9

VERACODE

The State of Software Security Today

We're living in an era where business competitiveness hinges on the speed and quality of software delivery. Some enterprises are struggling to keep up. Others are thriving.

Wherever they fall on that spectrum, all organizations race the competitive clock to deploy and evolve their game-changing applications. The question is, how well does application security keep up with it all?

This is the fundamental question Veracode asked this year as our team examined the data for *State of Software Security (SOSS) Volume 9*.

For a long time now, *SOSS* has provided a reliable yardstick for the most common vulnerabilities found in software, as well as how organizations are measuring up to security industry benchmarks throughout the software development lifecycle (SDLC). One thing we've always wanted to understand better, though, is how quickly these organizations are actually fixing flaws once they've been identified in application security scans.

This year, we turned our data analysis up a notch by working with the data scientists at Cyentia Institute, so that we could gain better visibility into the factors that go into fixing flaws. Readers will find valuable insight on how factors like flaw severity, business criticality of applications, and exploitability of the flaws change the rate at which certain vulnerabilities are fixed.

In many ways, our deeper look into the data confirmed what many industry veterans recognize intuitively: it takes time to fix security flaws. Contrary to what some security staffers might believe, developers simply can't wave a magic wand over the portfolio to fix the majority of flaws in an instant, or even in a week. On top of that, there are other factors at play, including QA, product release cycles, and other exigencies of delivering software to the real world.

However, our data presents hopeful glimpses at potential prioritizations and software development methods that could help organizations reduce risk more quickly. At the top of that list is the DevSecOps mentality, which tends to incorporate more frequent security scans, incremental fixes, and faster rates of flaw closures into the SDLC. This year's analysis shows a very strong correlation between high rates of security scanning and lower long-term application risks, which we believe presents a significant piece of evidence for the efficacy of DevSecOps.

Alongside that, we also offer up loads of valuable information about industry performance, third-party component risks, and vulnerability trends. We believe that this body of work offers security practitioners and developers alike valuable food for thought as they seek to improve their application security stance in the coming year.



LETTER FROM
Chris Eng
Vice President of
Research at Veracode

Sincerely, Chris Eng

Executive Summary

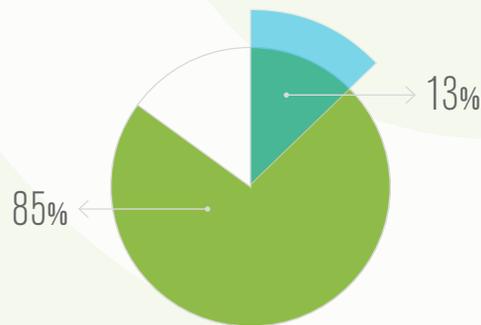
The metrics presented in Veracode's ninth iteration of the *State of Software Security (SOSS)* report represent the industry's most comprehensive set of application security benchmarks. Drawn from real-world applications, we have analyzed the data created through customer testing on Veracode's application security platform. It represents the scans of more than 2 trillion lines of code across 700,000 scans, all performed over a 12-month period between April 1, 2017 and March 31, 2018.

As in previous versions of the report, we'll provide insight into how well most applications adhere to industry best practices, like OWASP Top 10 guidelines, and which types of vulnerabilities turn up most in typical applications:

INDUSTRY BEST PRACTICE ADHERENCE

The pass rate for **OWASP Top 10** compliance on initial scan declined for the third year in a row, down to **22.5%**

More than 85% of all applications have at least one vulnerability in them; more than 13% have at least one critical severity flaw.



Close rates improved by 12 percentage points this year – customers closed almost **70%** of vulnerabilities they found.

Software is still rife with vulnerable components.

APPROXIMATELY



apps contain at least one vulnerable component.

THE MOST COMMON VULNERABILITIES PRESENT IN APPLICATIONS REMAINED LARGELY THE SAME:



SQL injection flaws are still present in nearly one in three applications.



Cross-Site Scripting (XSS) vulnerabilities are found in nearly 50% of applications.

→ As we worked on the report, we recognized that our data could provide even more insight than the standard benchmarks we've always analyzed in the past.

The most important function of an application security program is how effectively flaws are fixed once they are discovered. Our goal this year was to really delve deep into the statistics that show how long different types of vulnerabilities take to get fixed, and to understand why certain risks linger for as long as they do.

VULNERABILITY FIX BEHAVIORS

The data scientists at Cyentia Institute helped us to tell this story around vulnerability fix behavior. We were able to break down how different variables like flaw type, severity, app criticality, and rate of scanning impact the fix velocity and, conversely, the persistence of flaws once they've been discovered:

- More than 70% of all flaws remain 1 month after discovery and nearly 55% remain 3 months after discovery.
- 1 in 4 high and very high severity flaws are not addressed within 290 days of discovery.
- Flaws persist 3.5x longer in applications only scanned 1 to 3 times per year compared to ones tested 7 to 12 times per year.
- DevSecOps unicorns exist, and they greatly outperform their peers in how quickly they fix flaws; the most active DevSecOps programs fix flaws more than 11.5x faster than the typical organization.
- Infrastructure, manufacturing, and financial industries have the hardest time fully addressing found flaws.

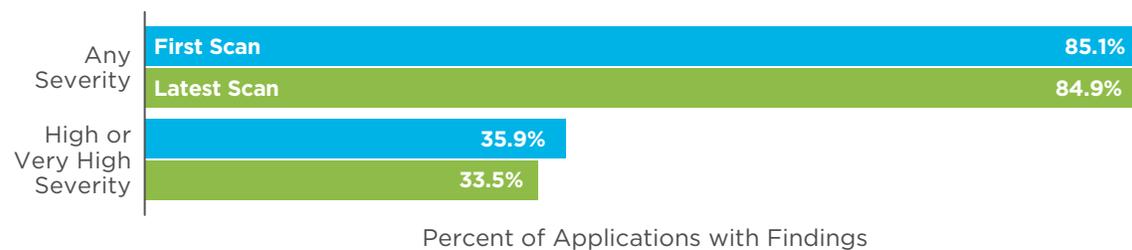
The data analysis tells some very important stories for security professionals and development teams alike about how they can take measurable steps to reduce application risks. We hope our readers are able to use all of these benchmarks to good effect. →

Overall State of Software Security

Our annual SOSS data puts hard evidence on the table to explain why so many security professionals experience anxiety when they think about application security (AppSec). There is no way to sugar coat it: the sheer volume of flaws and percentage of vulnerable apps remain staggeringly high.

In examining the data for the percentage of applications under test by our customers in the past year, we can see that the vast majority of them suffer from at least one vulnerability. A significant number of these vulnerabilities are of high or very high severity.

FIGURE 1: APPS WITH AT LEAST ONE VULNERABILITY

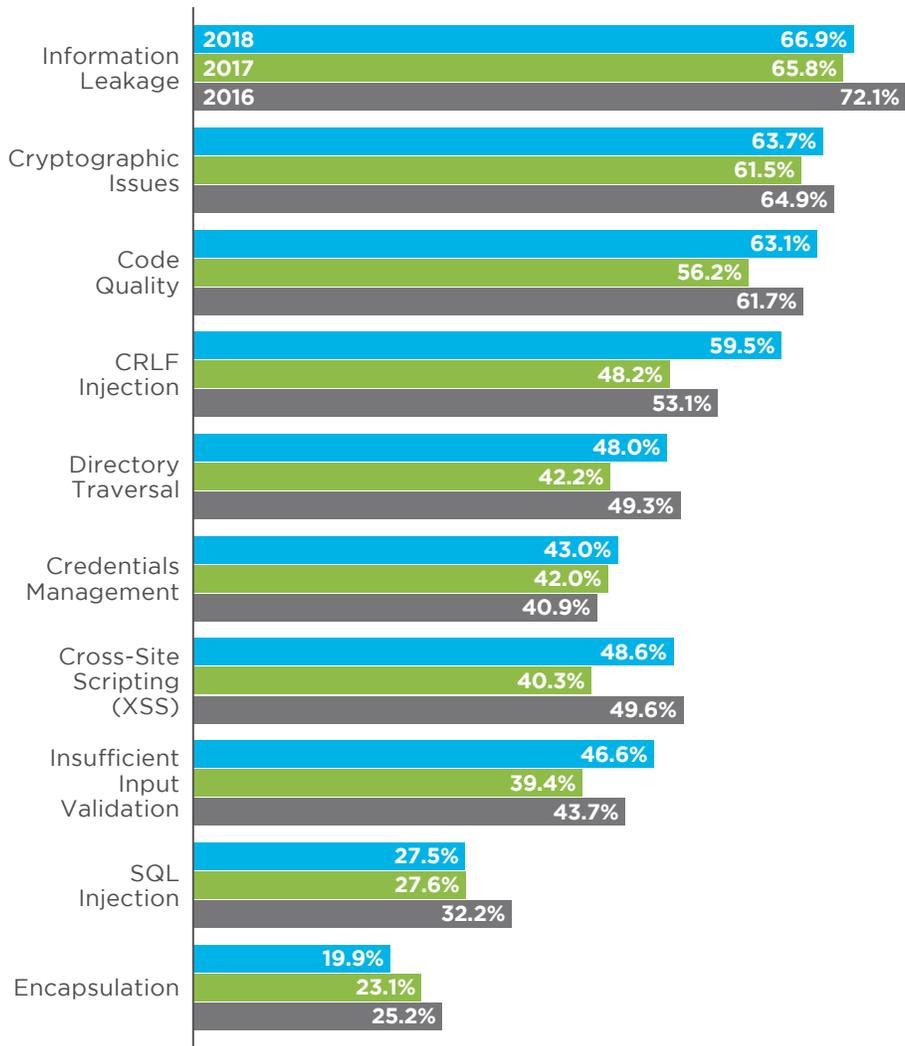


Source: Veracode SOSS Volume 9

Throughout the report, we share data from two types of scans. We commonly look at the first scan of applications, which indicate testing of applications that haven't previously gone through the AppSec program. We also look at latest scan statistics, which includes tests of applications that are currently in the middle of remediation and those applications for which organizations have deemed they've fixed enough flaws and have stopped scanning any further. Even on our customers' latest scans, we found that one in three applications were vulnerable to attack through high or very high severity flaws.

Breaking down the prevalence of flaws by vulnerability categories shows that all of the usual suspects are present at roughly the same rate as in previous years. In fact, our top 10 most prevalent flaw types have hardly budged in the past year.

FIGURE 2: PREVALENCE OF COMMON FLAW TYPES



Percent of Applications

Source: Veracode SOSS Volume 9, n=(2018:25.7k)

That means that organizations across the board have made very little headway to create awareness within their development organizations about serious vulnerabilities, like cryptographic flaws, SQL injection, and cross-site scripting. This is most likely a result of organizations struggling to embed security best practices into their SDLC, regardless of where the standards are from. The data shows that plainly here.

FIGURE 3: ADHERENCE TO INDUSTRY STANDARDS



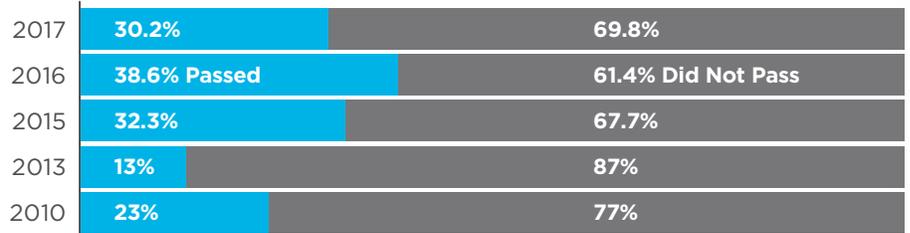
Passing Rate of Applications

Source: Veracode SOSS Volume 9
n=(First:30.8k, Latest:23.2k)

→ A historic look at OWASP compliance on first scan shows that this year's pass rate looks significantly better than five years ago. Unfortunately, the rate of OWASP compliance hit its peak in 2016. This year marks the third in a row that OWASP pass rates have declined. One variable to note is that OWASP updated its Top 10 list in 2017. While Veracode policy support wasn't fully updated until the end of the data window for SOSS Vol. 9, this could have been a factor in the pass rates declining this year. Shifts in focus on vulnerability types take a while to be implemented.

The big question, of course, is how effective are organizations at closing vulnerabilities once they've found them through our scans?

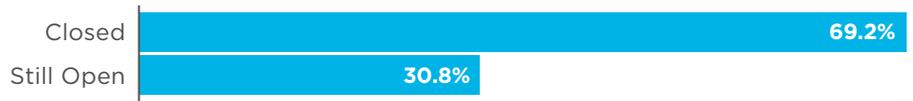
FIGURE 4: OWASP YEAR-BY-YEAR COMPARISON



Percentage of Applications Passing OWASP on First Scan
Source: Veracode SOSS Volume 9

The good news here is that customers are closing more of their flaws annually than in the past. Nearly 70% of flaws discovered in the past year were closed through remediation or mitigation - that's a jump of nearly 12 percentage points of closures since *State of Software Security Vol. 8*.

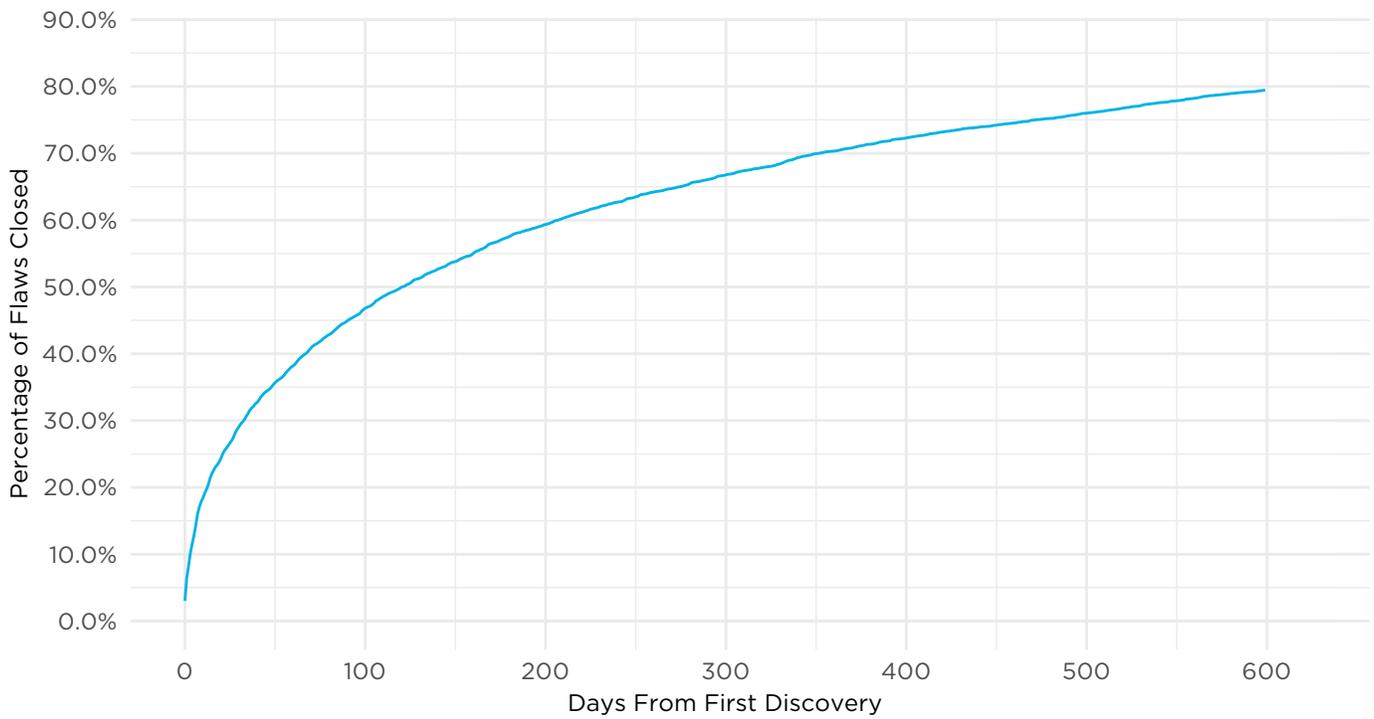
FIGURE 5: FLAWS CLOSED VS OPEN



Percent of Findings
Source: Veracode SOSS Volume 9, n=6.3m

Simply looking at the sheer volume of open to closed vulnerabilities only gives us so much visibility into the true efficacy of customers' AppSec practices. The time it takes for attackers to come up with exploits for newly discovered vulnerabilities is measured in hours or days. Which means that it is crucial to measure both how many flaws organizations close out every year, and how long it takes them to do so.

FIGURE 6: FIX VELOCITY



Source: Veracode SOSS Volume 9, n=6.3m

This year, we've taken a closer look at our customers' fix rate, and when we look at the curve for the average fix velocity from the first day of discovery, we see that it takes organizations a troubling amount of time to address most of their flaws. One week after first discovery, organizations close out only about 15% of vulnerabilities. In the first month, that closure reaches just under 30%. By the three-month mark, organizations haven't even made it halfway, closing only a little more than 45% of all flaws.

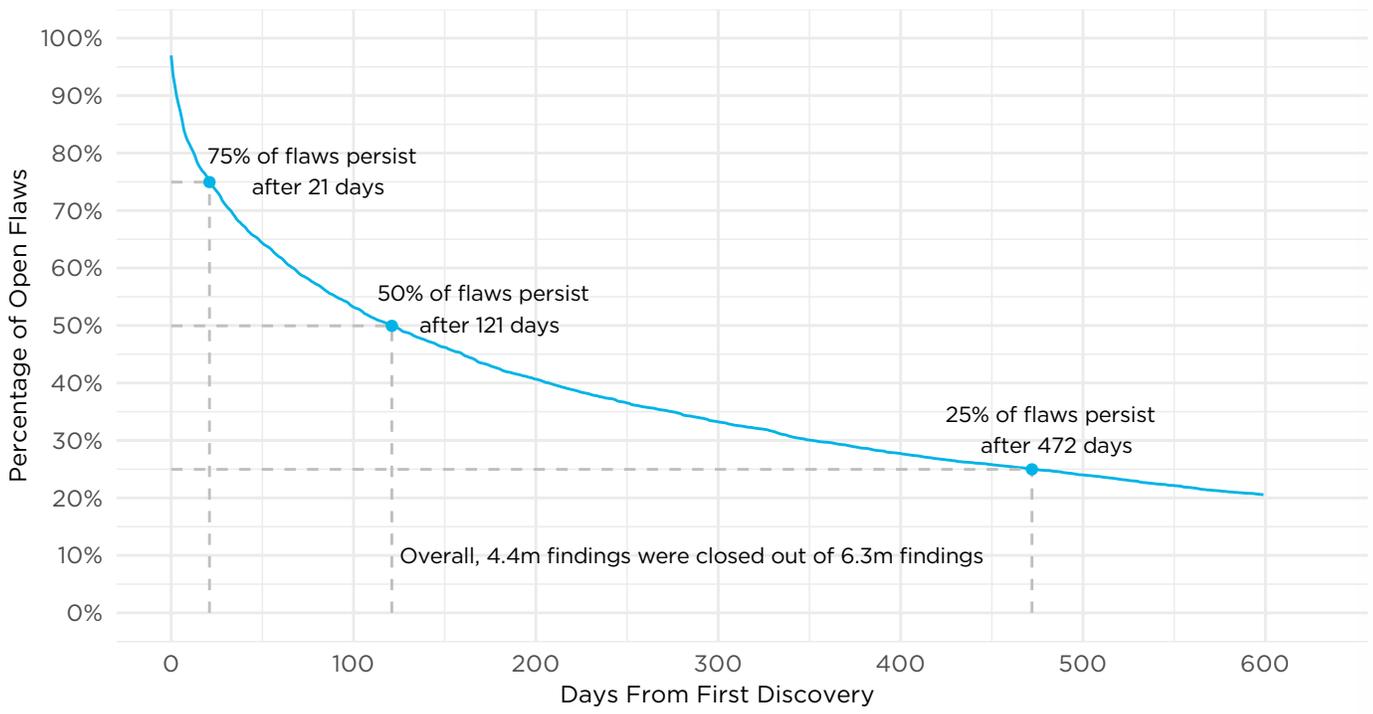


To put a finer point on this issue, the average velocity at which organizations are fixing flaws isn't just a mile marker for AppSec program performance — it's also a benchmark for measuring application risk.

Let's flip that curve and discuss the probability that a vulnerability will persist in an application over time.

We call this flaw persistence analysis.

FIGURE 7: FLAW PERSISTENCE ANALYSIS



Source: Veracode SOSS Volume 9, n=6.3m

Visualizing the data in this way allows us to get a clearer view of how long risk lingers in any given application under test. We've used flaw persistence as the basis for a lot of new investigation into this year's data. We hope this new view provides valuable insights into how customers prioritize the flaws they fix the fastest, as well as offering evidence of what isn't being fixed in a timely fashion, and how that impacts application risk exposure.

Focus on Fix

One thing is certain: the sheer volume of vulnerabilities present in most organizations' application portfolios makes it necessary for them to make daily tradeoffs between security, practicality, and speed. There are just too many vulnerabilities for organizations to tackle all at once, which means it requires smart prioritization to close the riskiest vulnerabilities first.

Remediation and mitigation of found vulnerabilities are the ultimate objective of Veracode customers, so we wanted to examine our data in a new way to give readers a better understanding of how organizations prioritize their fix behavior.

Understanding how long it takes to close vulnerabilities under different circumstances not only offers a glimpse into the current state of software security practices, but also highlights how organizations can work to incrementally improve their own security.

Understanding Flaw Persistence Intervals

In the previous section, we shared what we call flaw persistence analysis for all the applications our customers are testing. That analysis presents a line curve to show the probability that a vulnerability will remain in any given application over time, and we denoted the points in time on the curve at which 25%, 50%, and 75% of flaws in a typical application are usually fixed.

To better understand how long different kinds of flaws tend to linger in applications, we are using these percentiles to chart out what we call flaw persistence intervals. Below, you will see the flaw persistence interval for all applications, which corresponds to the flaw persistence analysis curve shown in the previous section.

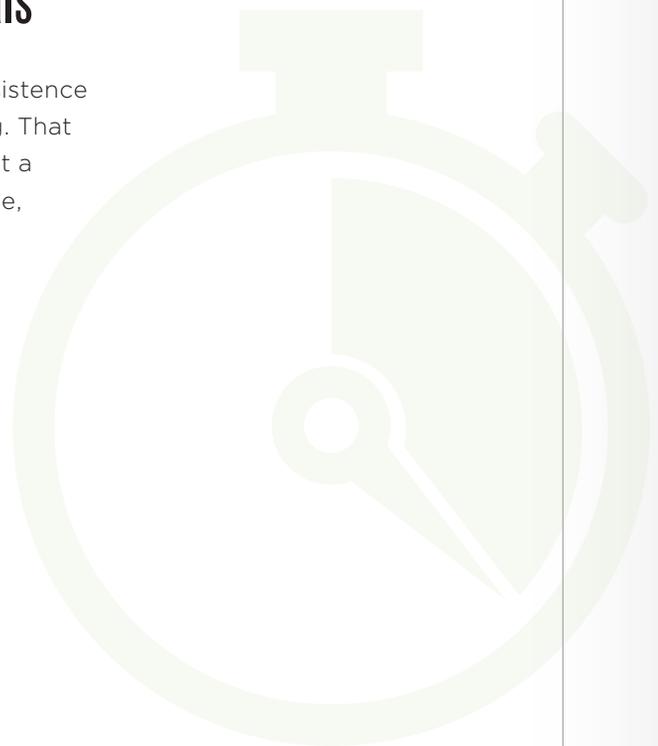
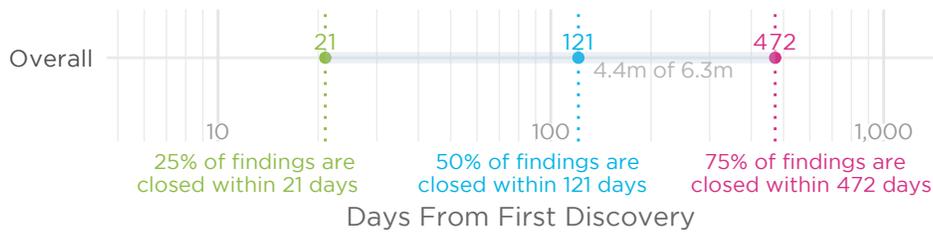


FIGURE 8: OVERALL FLAW PERSISTENCE INTERVAL



Source: Veracode SOSS Volume 9

In green, you will see that it takes 21 days to close 25% of vulnerabilities. In blue, the chart shows that it takes 121 days to close 50% of vulnerabilities. In pink, the data shows that it takes 472 days to close 75% of vulnerabilities. That means that, overall, one in four vulnerabilities remain open well over a year after first discovery.

This overall flaw persistence interval serves as the benchmark against which we will compare other intervals throughout the rest of the report. Readers should note that the dotted lines in green, blue, and pink on this and subsequent charts track to the plots on this first overall interval chart. This will provide visibility into whether certain factors correlate to a speeding up or slowing down of the rate of vulnerability closures compared to the overall norm. Interval plots to the left of a corresponding line indicate a faster speed in reaching that particular milestone, while plots to the right of the corresponding line indicate a slower speed of remediation.

One in four vulnerabilities remain open well over a year after first discovery.

Flaw Severity

Let's begin with one of the variables that application security teams are most urged to target for speedy remediation: vulnerability severity.

The potential impact to the confidentiality, integrity, and availability of the application determines the flaw severity of any given vulnerability. The highest severity flaws are less complicated to attack, offer more opportunity for full application compromise, and are more likely to be remotely exploitable — overall they tend to open up a wider attack blast radius.

SEVERITY SCORES ON OUR FIVE-POINT SCALE ARE RATED AS FOLLOWS:

| Severity Score | Description |
|----------------|---|
| 5 | Very High: The offending line or lines of code is a very serious weakness and is an easy target for an attacker. The code should be modified immediately to avoid potential attacks. |
| 4 | High: The offending line or lines of code have significant weakness, and the code should be modified immediately to avoid potential attacks. |
| 3 | Medium: A weakness of average severity. These flaws should be fixed in high assurance software. You should consider fixing this weakness after you fix the very high and high flaws for medium assurance software. |
| 2 | Low: This is a low priority weakness that will have a small impact on the security of the software. You should consider fixing these flaws for high assurance software. Medium- and low-assurance software can ignore these flaws. |
| 1 | Very Low: Minor problems that some high assurance software may want to be aware of. These flaws can be safely ignored in medium- and low-assurance software. This year's data found these flaws only in manual and dynamic scans — static data analyzed in this section does not include flaws in this severity level. |
| 0 | Informational: Issues that have no impact on the security quality of the application but which may be of interest to the reviewer. |

Breaking down the flaw persistence intervals based on where vulnerabilities fall on this scale shows that organizations are making a big push to fix their highest severity vulnerabilities first.

The first quartile of very high vulnerability closures is made more than a week sooner than the norm, and organizations managed to start working on the last quartile of very high vulnerabilities 237 days sooner than the norm. Though the intervals for burning down the first 25% and 50% of high severity flaws tracked with the norm, organizations managed to reach closure on 75% of these high severity flaws more than 100 days sooner than the norm.

On the flip side, low severity flaws were attended to at a significantly slower rate than the average speed of closure. It took organizations an average of 604 days to close three-quarters of these weaknesses.

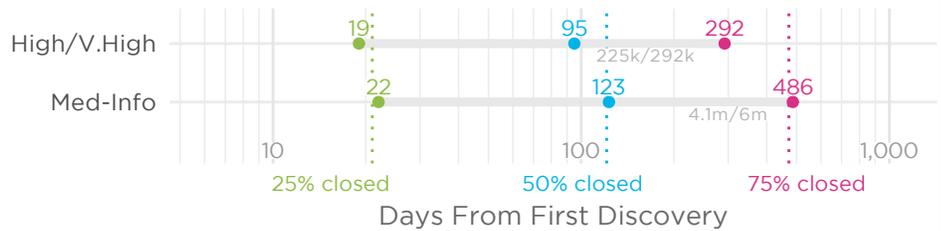
FIGURE 9:
FLAW PERSISTENCE INTERVALS BY FLAW SEVERITY



Source: Veracode SOSS Volume 9

In order to give a clearer picture of how severity prioritization is realistically working out in most situations, we rolled flaw persistence intervals into two severity groupings. The first group encompassed very high and high vulnerabilities, and the second included everything below that.

FIGURE 10:
SIMPLIFIED FLAW PERSISTENCE INTERVALS BY SEVERITY

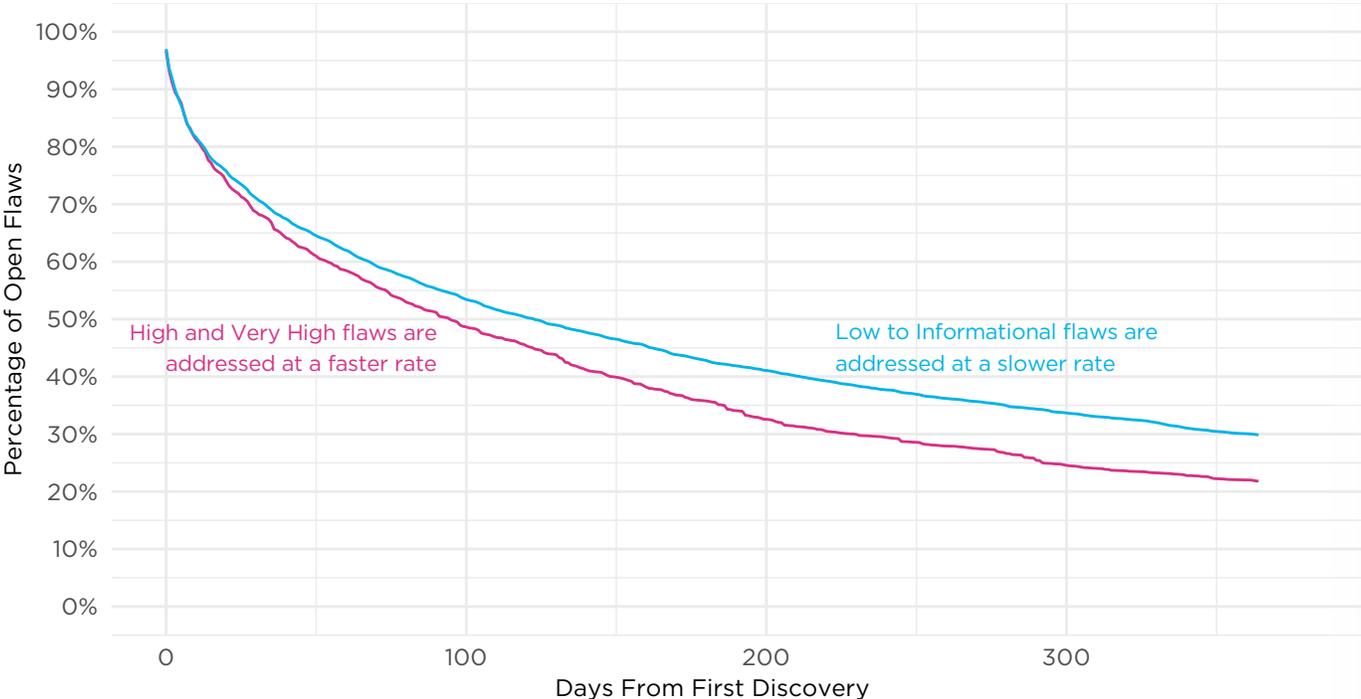


Source: Veracode SOSS Volume 9

This pair of intervals more clearly shows the correlation between the severity of the vulnerability and the speed of closure. Organizations hit the three-quarters-closed mark about 57% sooner for high and very high vulnerabilities than for their less severe counterparts.

If we translate the numbers into flaw persistence analysis curves, you can see even more clearly what the persistence delta looks like between the two severity clusters from the date of first discovery onward.

FIGURE 11: SEVERITY FLAW PERSISTENCE ANALYSIS



Source: Veracode SOSS Volume 9



Organizations hit the three-quarters-closed mark about **57%** sooner for high and very high vulnerabilities than for their less severe counterparts.



Exploitability

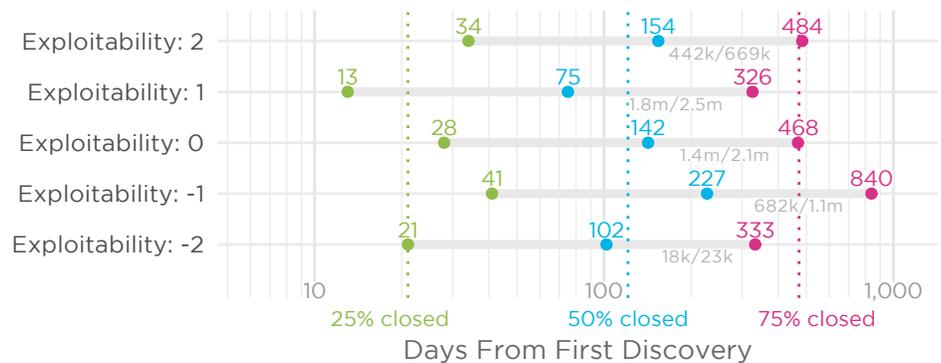
Exploitability adds another dimension to the measurement of the seriousness of a flaw. While severity scoring looks at a flaw through the lens of its potential overall impact on the application, exploitability specifically estimates the likelihood a flaw will be attacked based on the ease with which exploits can be executed. It is important to look at exploitability ratings to specifically prioritize those vulnerabilities that are both high impact and trivial to take advantage of. For example, a high severity flaw with a very high exploitability score introduces a lot more risk than a high severity flaw with a very low exploitability score.

When we examine the flaw persistence intervals based on exploitability, there are a few surprises that jump out at us. While the flaws judged as likely to be exploited with a score of “Exploitability: 1” have a sped-up flaw persistence interval relative to the average and to other lower exploitability scores, the next higher exploitability category does not. Those flaws ranked very likely to be exploited with an “Exploitability: 2” rating actually trail the average time for closure in all three of the flaw persistence intervals. It takes 40 days longer to close out 75% of these highly exploitable flaws than it does the average vulnerability.

VERACODE'S FLAW EXPLOITABILITY SCORING

| | |
|----|-----------------------------|
| 2 | Very likely |
| 1 | Likely |
| 0 | Neither likely nor unlikely |
| -1 | Unlikely |
| -2 | Very unlikely |

FIGURE 12: FLAW PERSISTENCE INTERVALS BY EXPLOITABILITY



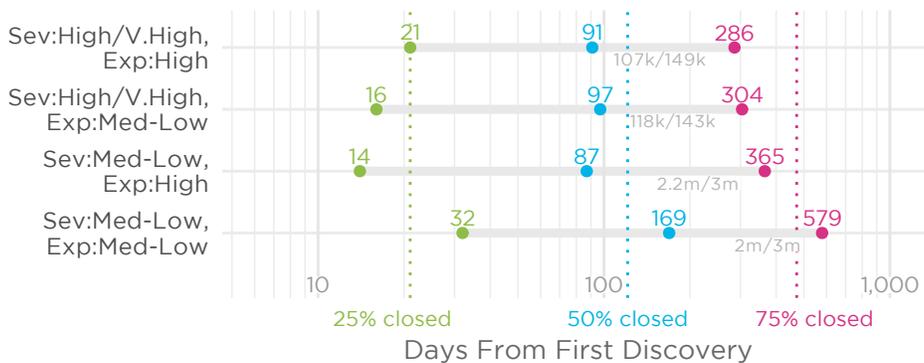
In order to get a clearer picture on how exploitability impacts remediation priorities within pools of similar severity flaws, we created additional flaw persistence intervals that analyzed different combinations of severity and exploitability. In these instances, we did see a few differentiations we'd expect to see. For example, for Severity 2 and 3 flaws, they were getting to the last quartile of open flaws a whopping 214 days faster when they were highly exploitable. But exploitability made a much less dramatic difference within the pool of Severity 4 and 5 vulnerabilities.

It is hard to tell exactly what is going on here with this counterintuitive result, but there are a few possibilities.

First of all, exploitability is more of a secondary prioritization metric than severity. Veracode typically recommends that developers use exploitability scoring as a way to sift through a cluster of vulnerabilities of a similar severity and ease of fix, putting the most exploitable of those on the top of that particular cluster.

We thought it could be that there were a number of highly exploitable but lower severity flaws that were skewing the flaw persistence intervals for this group - particularly considering that this category has a much smaller sample size than the other lower exploitability scores.

FIGURE 13:
FLAW PERSISTENCE INTERVAL BY SEVERITY AND EXPLOITABILITY



Source: Veracode SOSS Volume 9

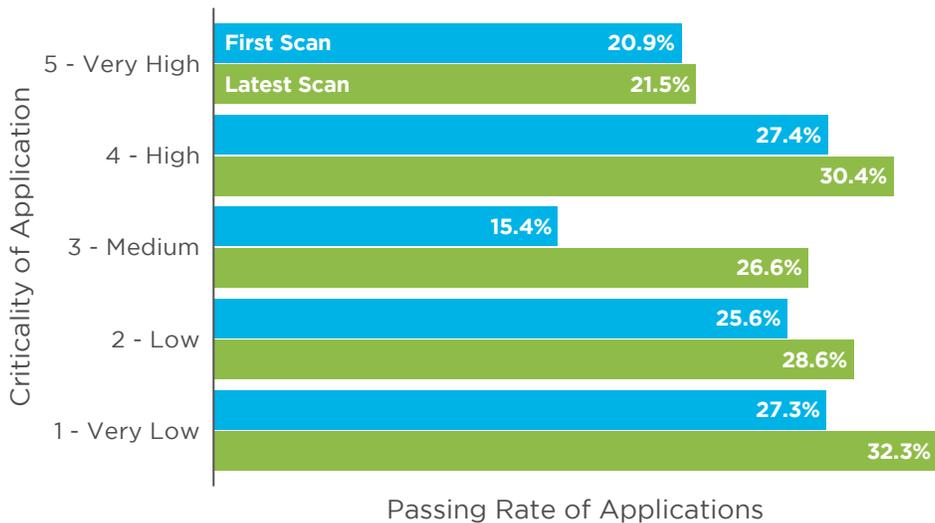
It could be that we're seeing another variable arising, namely the difficulty of remediation. The most severe and exploitable flaws are vulnerabilities deeply embedded in the underlying architecture of an application and require more complex remediation work. As such, they're much more difficult to fix and that could be what is extending flaw persistence in a population of flaws that should be at the very top of the priority list for remediation.

Application Criticality

In a textbook scenario, the properties of the vulnerability itself shouldn't be the only factors driving fix prioritization. A big part of the risk equation is the value of a particular asset at risk. As such, organizations should — in theory — also be weighting the business criticality of an affected application into their prioritization calculations.

However, when we looked at the data, we discovered that this is not happening to a very large degree. For example, a distribution of first scan and latest scan pass rates showed that the most important applications passed at a lower rate than other applications, and they didn't even show a higher improvement rate between first and latest scan compared to the others.

FIGURE 14: FIRST SCAN VS LATEST SCAN BY CRITICALITY OF APP

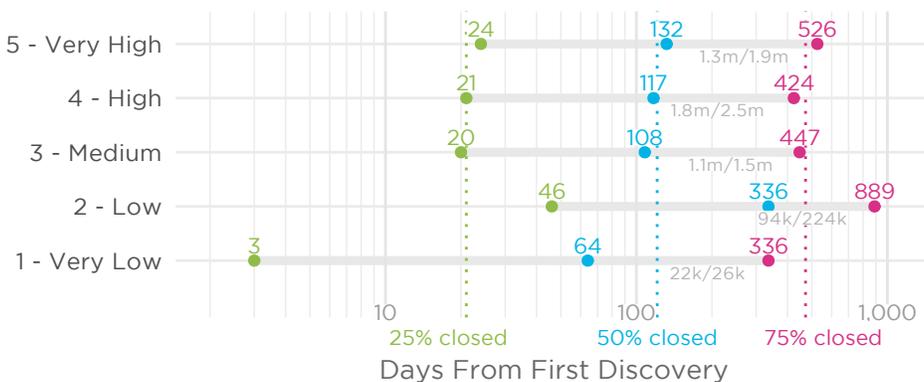


Source: Veracode SOSS Volume 9, n=53.4k

The data for flaw persistence based on business criticality further bore out our conclusion that organizations aren't using business criticality as a very strong prioritization variable.

While vulnerabilities in low criticality applications do trail all others in speed to reach all three closure percentiles, the flaws in very low criticality applications are addressed the quickest. This is a quirk of the data that we're trying to understand — it could be that the small sample size is adding greater variability into the findings.

FIGURE 15: FLAW PERSISTENCE INTERVAL BY APPLICATION BUSINESS CRITICALITY



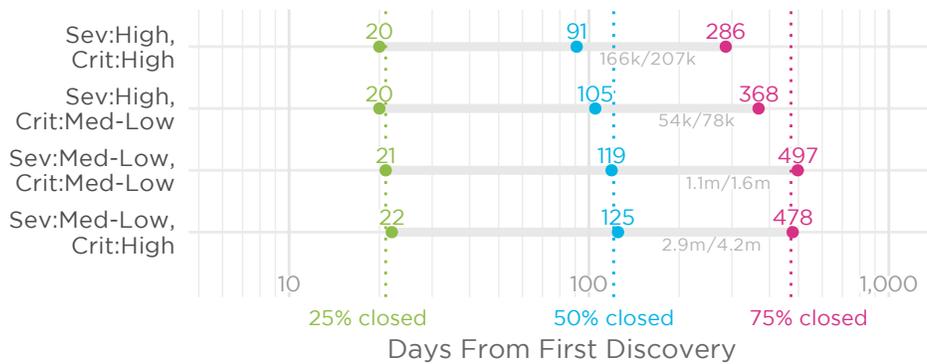
Source: Veracode SOSS Volume 9

What's more, the flaws in very high criticality apps are actually fixed more slowly than the average application. It takes well over two months longer to fix 75% of vulnerabilities in these mission-critical apps than it takes to reach the same mark in the average application.

Now, it is likely that the stability concerns and change management policies on mission-critical apps are much more stringent, which is likely impacting how quickly teams can get remediations deployed. But the lesson here is that these unfixed flaws are leaving extraordinary windows of risk open within organizations' most valuable application assets.

Drilling down further into the data, we can see that the disregard for app criticality mostly plays out even when filtered by severity of flaw.

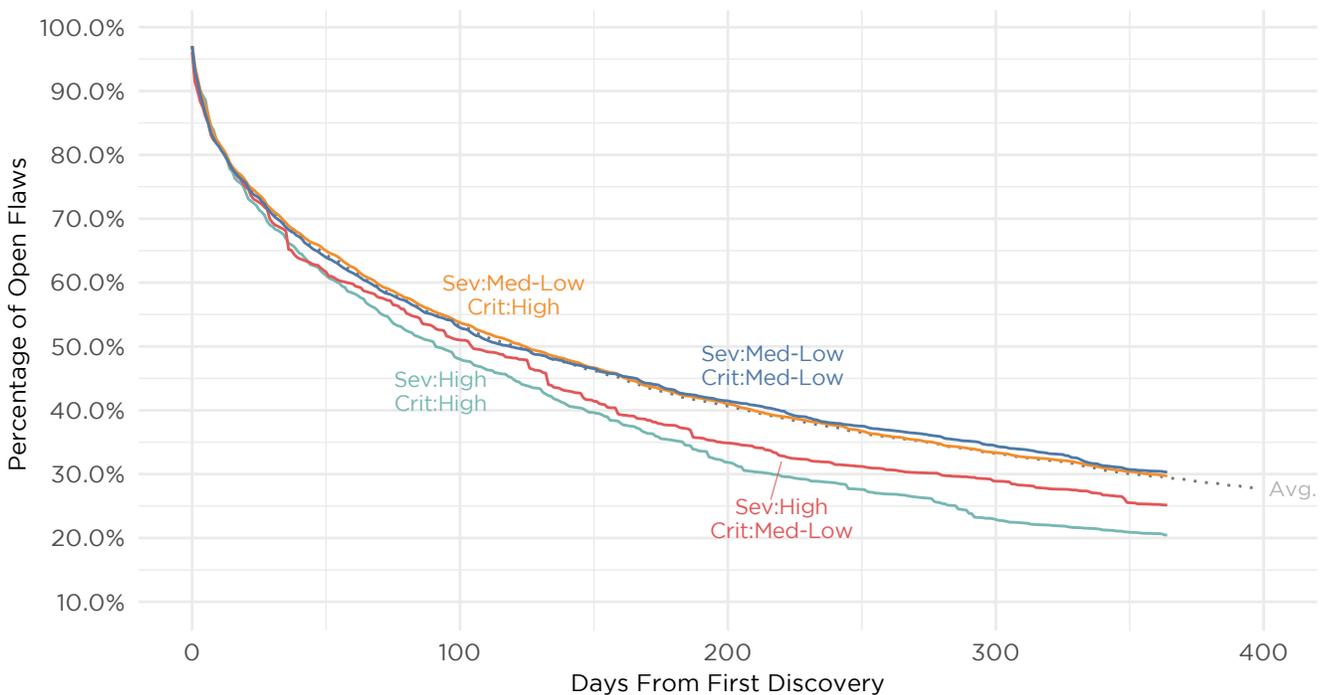
FIGURE 16: FLAW PERSISTENCE INTERVAL BY CRITICALITY AND SEVERITY



If we compare the flaw persistence analysis curves for groups paired by different criticality and severity scores, we see that they're more likely to be pulled by the severity of the flaw than the criticality of the app.

Source: Veracode SOSS Volume 9

FIGURE 17: FLAW PERSISTENCE ANALYSIS BY CRITICALITY AND SEVERITY



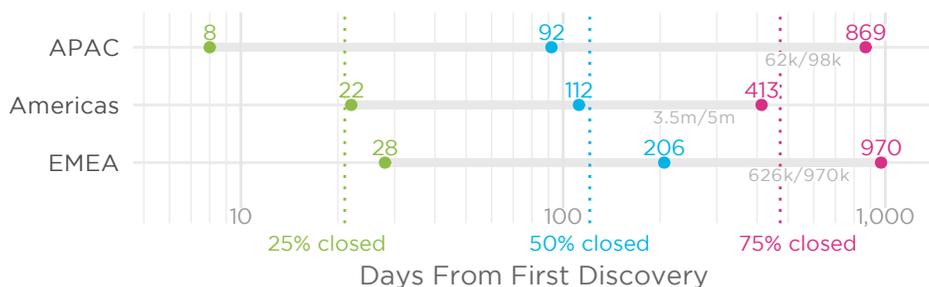
Source: Veracode SOSS Volume 9

The one silver lining to this occurs as organizations get toward the end of flaw burndown. It does seem like some prioritization kicks in to differentiate between the lingering highest vulnerability flaws that need to be addressed. Around the six-month mark, you can see a clear difference between the highest severity flaws in highly critical apps versus less important apps.

Regional Breakouts

While the Americas — particularly the U.S. — dominate the sample sizes, we were able to glean some insights into variations in flaw persistence based on regional differences.

FIGURE 18: FLAW PERSISTENCE INTERVAL BY REGION



Source: Veracode SOSS Volume 9

Unsurprisingly, vulnerabilities addressed by organizations in the Americas mostly tracked to the overall average. This was inevitable due to the fact that the large volume of these vulnerabilities weighted the average.

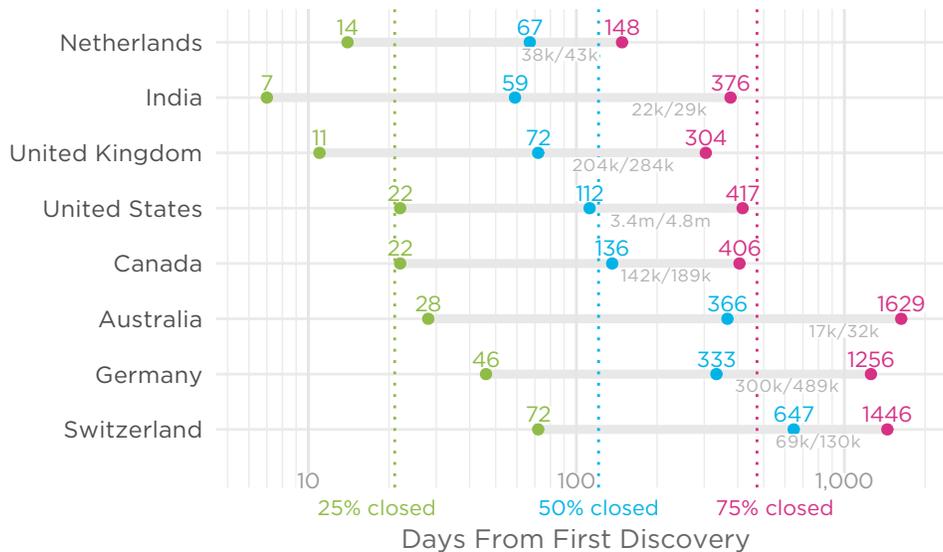
However, one thing to note is that companies in the Americas did outperform the average on the tail-end of the vulnerability burndown process. This indicates how badly companies in APAC and EMEA trailed when it came to getting to their last quartile of open vulnerabilities.

In examining the APAC companies' speed of closure, it is interesting to find that these firms jumped on their first chunk of flaws very quickly. It only took APAC companies about a week to close out 25% of their flaws. However, the spread between reaching that first milestone and eventually resolving 75% of flaws was enormous. It took APAC companies well over two years to start working on their last quartile of open vulnerabilities.

Meanwhile, EMEA companies lagged behind the average significantly at every milepost of the flaw persistence intervals. It took more than double the average time for EMEA organizations to close out three-quarters of their open vulnerabilities. Troublingly, 25% of vulnerabilities persisted more than two-and-a-half years after discovery.

Further breaking these persistence intervals out by country, we did find some regional outliers worth noting.

FIGURE 19: FLAW PERSISTENCE INTERVAL BY COUNTRY



Source: Veracode SOSS Volume 9

For example, companies in India, the United Kingdom, and the Netherlands greatly outperformed their regional counterparts in speed of fix.

In particular, the rapid rate of remediation evidenced by Dutch companies remain a promising bright spot amid the worrying time it took their EMEA counterparts to fix the same percentage of flaws. Dutch firms managed to start working on their last quartile of open flaws within five months of discovery — that is the fastest rate worldwide and three times as fast as the average application.

That sense of urgency was contrasted by outliers on the other end of the spectrum in Germany and Switzerland. It took German firms more than three years to reach their final quartile of open vulnerabilities, and it took Swiss organizations nearly four years to reach the same milestone.

Industry Breakouts

We will dive into industry benchmarks more fully later on in the report, but we would be remiss in discussing overall flaw persistence trends without touching on industry breakouts.

FIGURE 20: FLAW PERSISTENCE INTERVALS BY INDUSTRY



Source: Veracode SOSS Volume 9

Healthcare organizations are remediating at the most rapid rate at every interval compared to their peers. It takes just a little over seven months for healthcare organizations to reach the final quartile of open vulnerabilities, about eight months sooner than it takes the average organization to reach the same landmark. Similarly, retail and technology firms outpace the average speed of fix at every interval.

While infrastructure firms address the first half of their open flaws more rapidly than average, it takes them significantly more time to get to the second half. At least one in four vulnerabilities are left open almost three years after first discovery within infrastructure industry apps. This likely reflects the great difficulty that these firms face in fixing many applications within critical systems that have extremely tight thresholds for uptime and availability.

In a mirror to infrastructure situations, government and education firms have a reverse situation. They're right about on par with the average time to address the first half of their open flaws, but they start to pick up speed once they get over that hump. This could be an indication of bureaucratic inertia that may impede initial progress, but which is likely overcome once security teams and developers cut through the red tape.

DEFINITIONS

Remediating a flaw:

Changing the code to address the risk

Mitigating a flaw:

Documenting a compensating control that adequately addresses the risk associated with it

Remediation vs Mitigation

As we ruminate over the speed at which organizations are addressing vulnerabilities, it's worth taking a quick look at how these flaws are being closed out. In tracking flaw closures, there are two main categories — remediation and mitigation.

FIGURE 21: MITIGATION VS. REMEDIATION



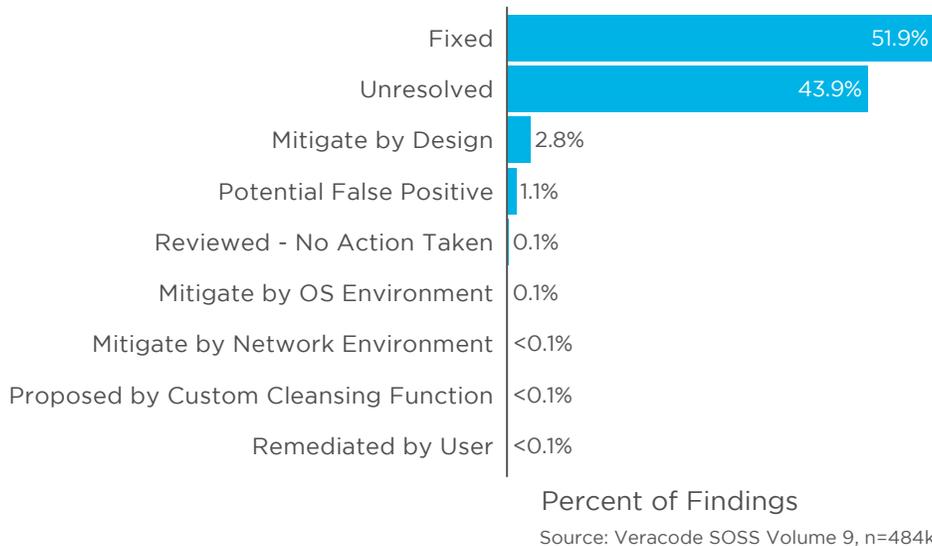
Source: Veracode SOSS Volume 9, n=484k

As we see here, a little over half of all flaws are fixed, and just under 44% of them are left open. Then there's a small sliver left over that are not closed out with a code fix but instead through mitigating factors noted by developers. This could be because developers deem them false positives, because they believe other elements of the application's design or its environment counterbalance the risk of the flagged vulnerability.

The good news here is that developers are clearly taking static application security tests seriously — they're not just blindly rejecting findings as false positives and moving on. In fact, all mitigation reasons account for a little more than 4% of vulnerability closures.

If we zoom in on just the vulnerabilities closed by mitigation, we can get an even clearer picture of the reasons noted by developers for closing out flaws without altering code.

FIGURE 22: DEVELOPER MITIGATION REASONS



This chart shows that potential false positives aren't even the first reason named by developers for a close by mitigation. In the majority of instances, developers accept that static analysis may be finding something in the application, but they disagree with the analysis on the assumptions made about the design or the environment to flag something as a flaw. This is where mitigation by design or by environment kicks in. While some of the assumptions developers are making to deem a flaw as mitigated may be up for debate in terms of how sound they really are, the good news is that these mitigations make up such a slim number of flaw closures. This should give organizations peace of mind that when a flaw is closed, it is either fixed or closed for good reasons.

Concluding Thoughts on Fixing Flaws

One final thought on the prioritization of how organizations fix flaws is that the flaw persistence intervals above do not really delve into the impact of policy on timing. Usually, individual organizational policies will drive our customers' fix behavior above all other factors, and each of those policy sets are unique. Based on our analysis, many policies clearly take into account flaw severity. Some might take into account exploitability, others might emphasize certain vulnerability categories, and a few others will dictate how fixes are made to specific applications based on what they do for the business.

At the end of the day, an individual developer is going to be looking at his or her organization's policy to chart the plan of attack for closing out vulnerabilities. For any given customer, those policies may be based on some of the variables we laid out here, or they could be based on other factors unique to their organization or industry.

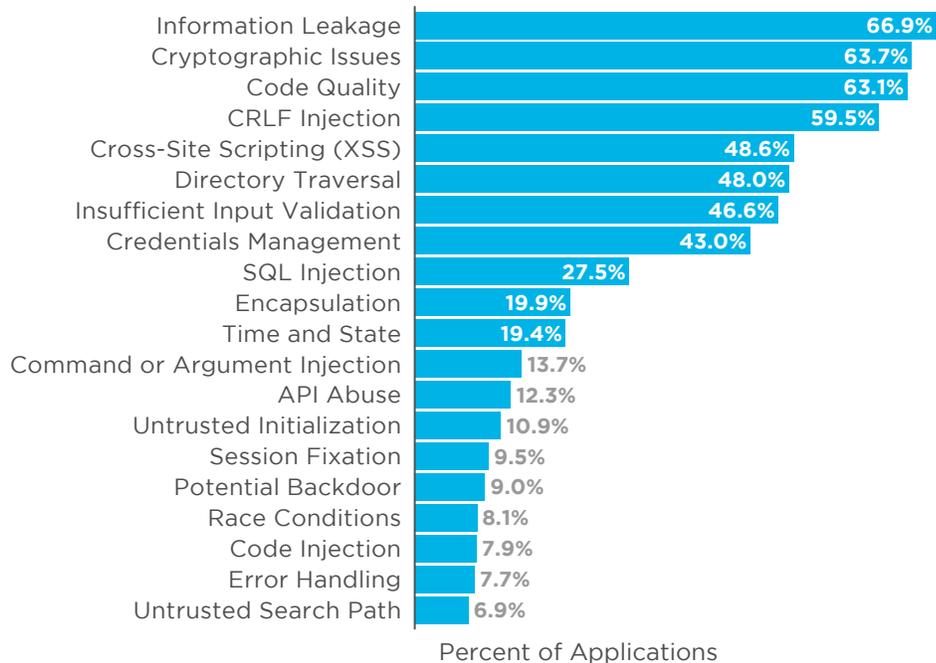
The takeaway for the data laid out in this section of *SOSS Vol. 9* is that organizations need to start thinking more critically about the factors that impact what they fix first. We called the charts laid out in this analysis flaw persistence intervals because we want to emphasize that they're offering a very detailed picture of the time of exposure faced by allowing these clusters of open vulnerabilities to linger.

Common Vulnerability Types

Overall Category Numbers

In analyzing the data, we found that the most common types of vulnerabilities cropped up in largely the same proportions as last year. The top four vulnerability categories presented themselves in more than half of all tested applications. This means the majority of applications suffered from information leakage, cryptographic problems, poor code quality, and CRLF Injection.

FIGURE 23: 20 MOST COMMON VULNERABILITY CATEGORIES



Source: Veracode SOSS Volume 9, n=25,790

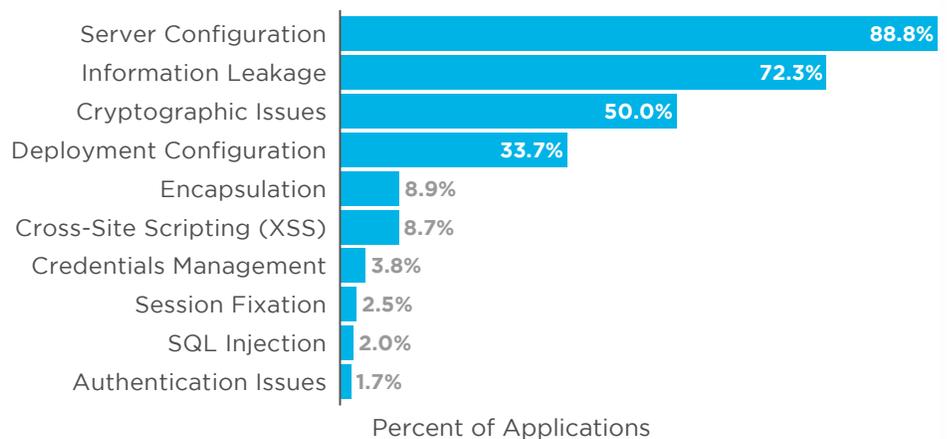
Other heavy-hitters also showed up in statistically significant populations of software. For example, we discovered highly exploitable cross-site scripting flaws in nearly 49% of applications, and SQL injection appeared nearly as much as ever, showing up in almost 28% of tested software.

One thing to keep in mind is that this particular distribution of common vulnerabilities was found through Static Analysis Security Testing (SAST), which examines code in a non-runtime environment. We've largely focused our data analysis on SAST results because we believe it is more statistically reflective of the high-level efficacy of AppSec during the SDLC. Static testing is more commonly done earlier in the SDLC, whereas dynamic tests are done later in the lifecycle for a variety of reasons, including the length of time it takes to test dynamically.

However, we should note that there are some differences in the occurrence of flaw types when we look at the prevalence in results for Dynamic Analysis Security Testing (DAST), which examines the application as it executes in a runtime environment.

Dynamic testing offers a totally different testing methodology and environment, so it shouldn't be surprising that it's stronger at dredging up different classes of flaws. The top 10 common vulnerabilities uncovered by DAST are still heavy on flaws like information leakage and cryptographic issues, but it also shows a higher prevalence of server configuration and deployment configuration flaws. These are flaws that simply can't be found prior to code execution, but which offer a very viable path to attack. As such, they still need to be on the AppSec radar.

FIGURE 24: TOP 10 VULNERABILITY CATEGORIES BY DYNAMIC APPLICATION SECURITY TESTING



Source: Veracode SOSS Volume 9, n=22,558

As we examine the top vulnerabilities, it is also crucial to consider that not every flaw type is created equal. It would be myopic to make judgements on risk simply by looking at flaw categories by volume of vulnerabilities present. For example, code quality flaws may be present in twice as many applications as SQL injection vulnerabilities, but that does not mean they pose twice as much risk as SQLi to the state of software security. Probably quite the opposite. As a class, SQLi tends to present flaws of a much higher severity and exploitability than code quality vulnerabilities.

Once organizations dig into individual vulnerabilities, they'll see that each of these category types exhibit different envelopes of risk based on exploitability and severity ratings. That must be taken into account when setting remediation priorities. However, even exploitability and severity metrics are not perfect indications of how to prioritize remediation of different flaw categories. Certain categories that may have relatively low measurements of severity or exploitability could hold significant risk in many situations — particularly when chained to other flaws. The key thing to keep in mind is context.

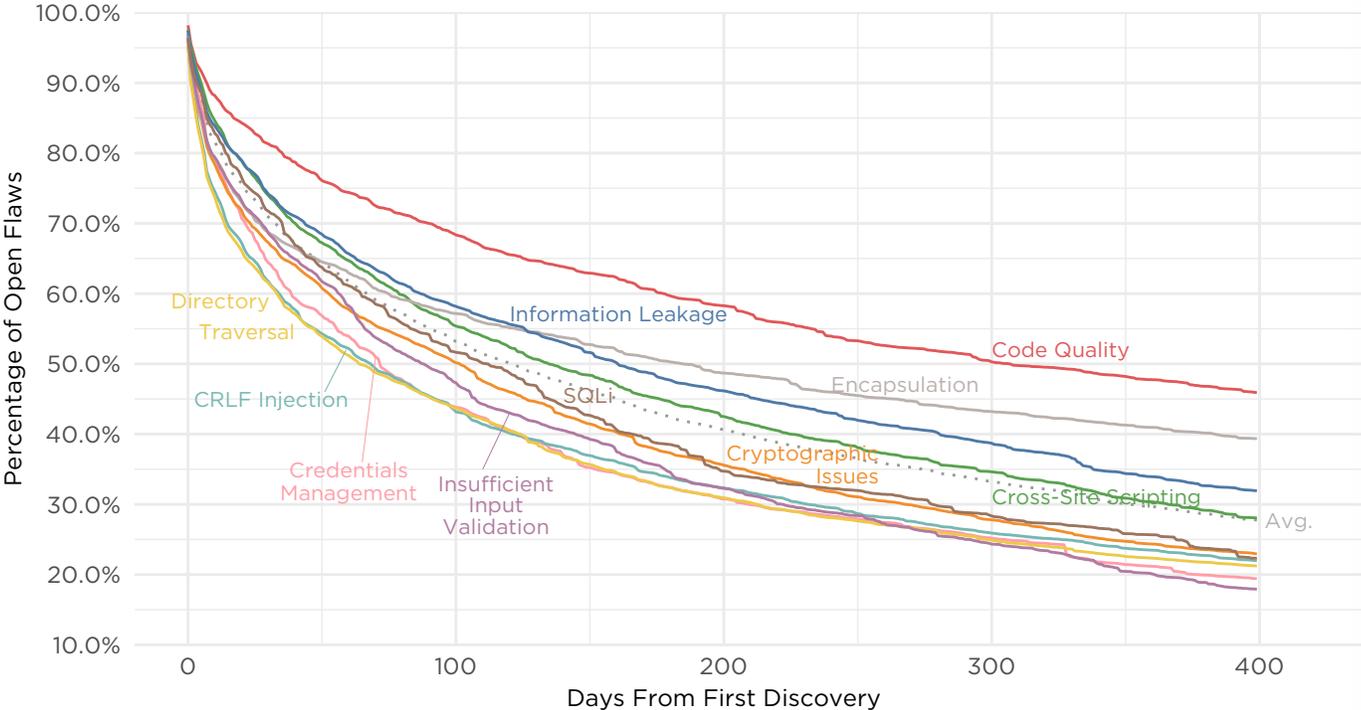
A low severity information leakage flaw could provide just the right amount of system knowledge an attacker needs to leverage a vulnerability that might otherwise be difficult to exploit. Or a low severity credentials management flaw, which might not be considered very dangerous, could hand the attackers the keys to an account that could be used to attack more serious flaws elsewhere in the software.



A low severity information leakage flaw could provide just the right amount of system knowledge an attacker needs to leverage a vulnerability that might otherwise be difficult to exploit.

Toxic combinations of flaws are not necessarily reflected in severity or exploitability ratings. In the real world, attack chaining matters. Being mindful of that reality adds further texture to the idea of flaw persistence. The more vulnerabilities organizations leave open to accumulate alongside other persistent flaws, the more attack surface the bad guys have to work with when stringing together their exploits.

FIGURE 25: FLAW PERSISTENCE ANALYSIS



Source: Veracode SOSS Volume 9

As we examine the flaw persistence of common flaw categories, we can easily see that each of these tracked flaw categories presents its own unique remediation challenges. Some of the deltas here in flaw persistence are simply reflecting the difference in severity of each flaw type. But certain flaw categories are also easier to fix than others, contributing to the sometimes wide differences in the time it takes to address some categories over others.

BREAKDOWN:

Top 10 Most Common Vulnerabilities

Information Leakage

FIGURE 26: INFORMATION LEAKAGE SNAPSHOT

Prevalence Rank
1

Prevalence
67%
of applications

Remediation Rank
8



These are flaws that allow the application to reveal sensitive data about the application, environment, or user, that could be leveraged by an attacker to hone future attacks against the application. These flaws are usually very low on the exploitability and severity ratings, but they frequently provide valuable breadcrumbs to attackers scoping out targets. They can be used to provide system and configuration information about victims so that attackers can target exploits specific to the victim's setup. Not to mention, data leaked through these vulnerabilities could be highly sensitive — directly leading to a high-profile data breach without any further attacks necessary.

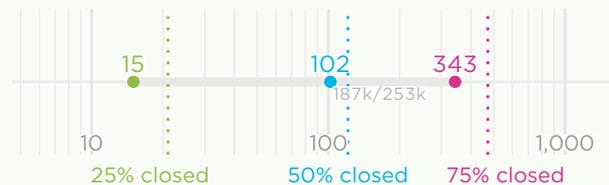
Cryptographic Issues

FIGURE 27: CRYPTOGRAPHIC ISSUES SNAPSHOT

Prevalence Rank
2

Prevalence
64%
of applications

Remediation Rank
5



This includes a number of risky cryptographic practices, including using broken crypto algorithms, improperly validating certificates, storing sensitive information in cleartext, and employing inadequate encryption strength. The flaw severity of these attacks is exclusively at three; nevertheless, these flaws are very serious. They may not necessarily lead to remote code execution, but they do very frequently lead to embarrassing and costly data breaches.

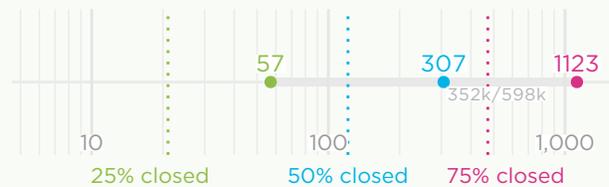
Code Quality

FIGURE 28: CODE QUALITY SNAPSHOT

Prevalence Rank
3

Prevalence
63%
of applications

Remediation Rank
10



These are common issues in code quality that could eventually impact the security of the application. Some examples include improper resource shutdown or release, leftover debug code, and using the wrong operator when comparing strings.

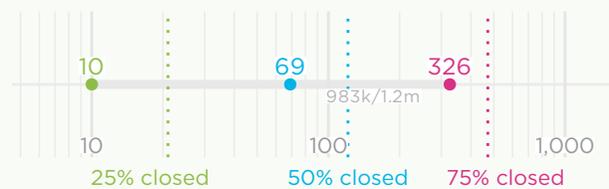
CRLF Injection

FIGURE 29: CRLF INJECTION SNAPSHOT

Prevalence Rank
4

Prevalence
60%
of applications

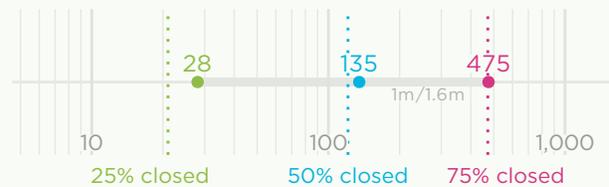
Remediation Rank
4



This includes any vulnerability that enables carriage return line feed (CRLF) injection attacks. Included here are flaws involving improper output neutralization for logs, and improper neutralization of CRLF in HTTP headers. These flaws are not rated high or critical, but they are generally pretty exploitable. They tend to lead to HTTP response splitting attacks, which are often then chained into XSS attacks.

Cross-Site Scripting (XSS)

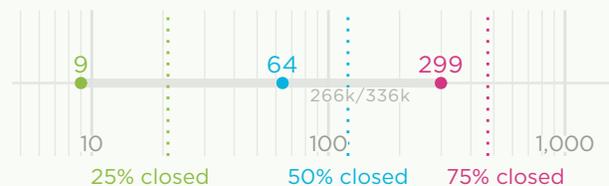
FIGURE 30: XSS SNAPSHOT



These are vulnerabilities that give attackers the capability to inject client-side scripts into the application, potentially bypassing security controls in the process. While XSS flaws are typically of moderate severity, these are some of the most exploitable flaws among the categories tracked. Unsurprisingly, they are also the number one favorite vulnerability type leveraged by attackers on the web today.

Directory Traversal

FIGURE 31: DIRECTORY TRAVERSAL SNAPSHOT



These flaws open up the possibility of attacks that give malicious actors the capability to gain unauthorized access to restricted directories and files. Like XSS attacks, directory traversals may only be moderately severe, but they are usually very exploitable. They are frequently chained-in attacks. This year, for example, researchers showed that it was possible to chain together attacks on several directory traversal vulnerabilities, combined with a few other flaws, in order to completely compromise a popular enterprise CRM system.

Insufficient Input Validation

FIGURE 32: INSUFFICIENT INPUT VALIDATION SNAPSHOT



Tainted input is the root cause of many security headaches. This category includes a number of input validation flaws that open up the application to malformed input that can cause security issues. This includes vulnerabilities involving open redirect and unsafe reflection.

Credentials Management

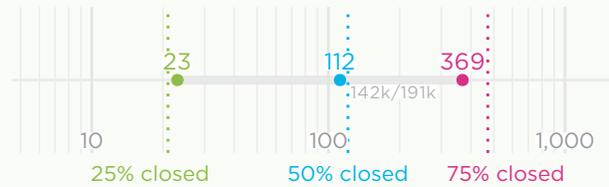
FIGURE 33: CREDENTIALS MANAGEMENT SNAPSHOT



These are errors in the handling of user credentials that can enable attackers to bypass access controls. Some of the most common errors include hard-coded passwords and plaintext passwords in configuration files and elsewhere. These flaws are often scored with a low severity rating that does not indicate the true seriousness of these flaws. Something like a hard-coded password can easily provide the keys to the kingdom if an attacker has some knowledge of the system that the victim uses. For example, in commercial software, attackers may glean that knowledge simply by reading a manual.

SQL Injection

FIGURE 34: SQL INJECTION SNAPSHOT



One of the most severe categories of this group, SQLi are any vulnerability that allow the attacker to gain unauthorized access to a back-end database by using maliciously crafted input. They are almost exclusively Severity 4 flaws with extremely high exploitability ratings. According to their flaw persistence intervals, organizations leave one in four of these vulnerabilities open for more than a year after discovery. These are behind only XSS in terms of flaws most exploited on the web.

Encapsulation

FIGURE 35: ENCAPSULATION SNAPSHOT



These vulnerabilities involve code that does not sufficiently encapsulate critical data or functionality. This includes trust boundary violations, protection mechanism failures, and deserialization of untrusted data.

Snapshots of Five Other Serious Flaw Categories

As we alluded to in our overview of common flaw categories, the frequency of flaw occurrence does not give the most complete picture of the overall risk profile of security today. As such, we would like to highlight a few other serious flaw categories that did not make the top 10 list – but are worth tracking because of how severe and/or exploitable they truly are.

FIGURE 36: COMMAND OR ARGUMENT INJECTION SNAPSHOT

Prevalence Rank
12

Prevalence
14%
of applications

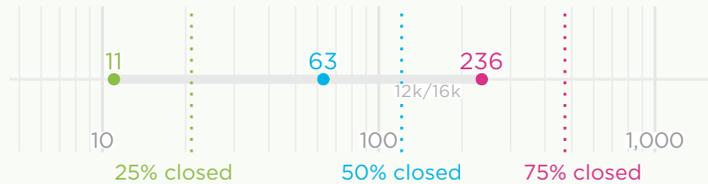


FIGURE 37: BUFFER OVERFLOW SNAPSHOT

Prevalence Rank
25

Prevalence
3%
of applications

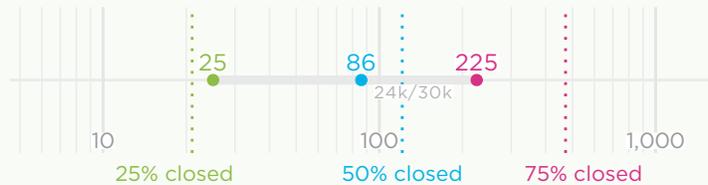


FIGURE 38: DANGEROUS FUNCTIONS SNAPSHOT

Prevalence Rank
27

Prevalence
2%
of applications

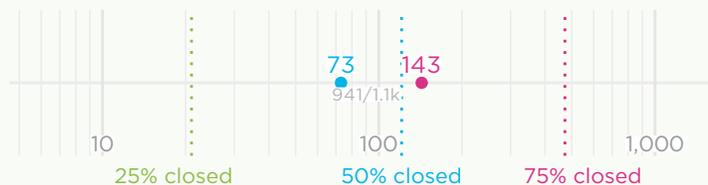


FIGURE 39: UNTRUSTED INITIALIZATION SNAPSHOT

Prevalence Rank
14

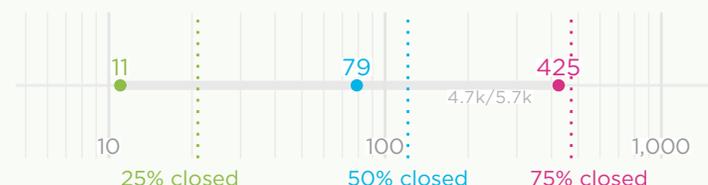
Prevalence
11%
of applications



FIGURE 40: UNTRUSTED SEARCH PATH SNAPSHOT

Prevalence Rank
20

Prevalence
7%
of applications



The DevSecOps Effect

DevOps practices have taken the IT world by storm. Enterprises increasingly recognize that the speed of software delivery spurred on by DevOps practices can often be a game changer when it comes to digital transformation and business competitiveness. One study by CA Technologies recently showed that the highest performing organizations in DevOps and Agile processes are seeing a 60% higher rate of revenue and profit growth, and are 2.4x more likely than their mainstream counterparts to be growing their business at a rate of more than 20%.



As the DevOps movement has unfolded, security-minded organizations have recognized that embedding security design and testing directly into the continuous software delivery cycle of DevOps is a must for enterprises. This is the genesis of DevSecOps principles, which offer a balance of speed, flexibility, and risk management for organizations that adopt them. The difficulty is that, until now, it has been tough to find concrete evidence of DevSecOps' security benefits.

That's all changing, because we've made some significant breakthroughs with our SOSS 9 analysis.

This is the third year in a row that we've documented momentum for DevSecOps practices in the enterprise, and now with our flaw persistence analysis, we've also got hard evidence to show that DevSecOps has the potential to be a very positive influence on the state of software security.

Our data shows that customers taking advantage of DevSecOps' continuous software delivery are closing their vulnerabilities more quickly than the typical organization.



Scan Frequency and Cadence

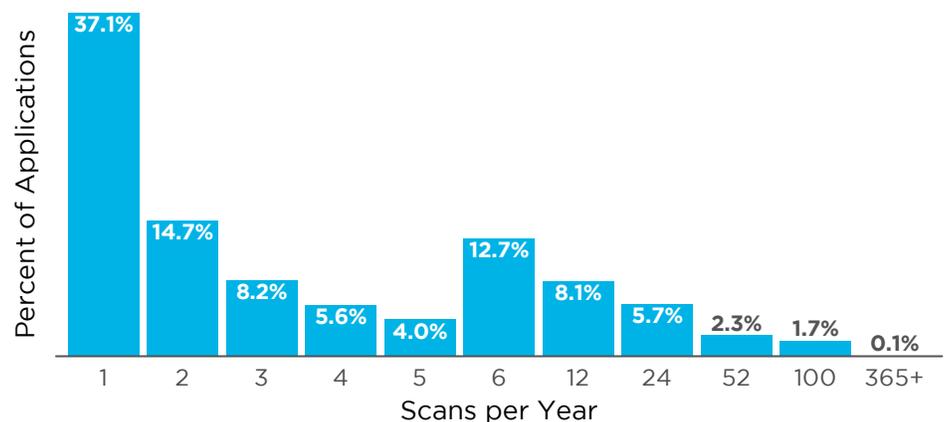
Over the past three years, we’ve examined scanning frequency as a bellwether for the prevalence of DevSecOps adoption in our customer base. Our hypothesis is that the more frequently organizations are scanning their software, the more likely it is that they’re engaging in DevSecOps practices.

Incrementalism is the name of the game in DevOps, which focuses heavily on deploying small, frequent software builds. Doing it this way makes it easier to deliver gradual improvements to all aspects of the application. When organizations embrace DevSecOps, they embed security checks into those ongoing builds, folding in continuous improvement of the application’s security posture alongside feature improvement.

Keeping this in mind, it’s only natural that a DevSecOps organization will scan much more frequently than a traditional waterfall development organization. These organizations tend to top-load huge changes into a lengthy development cycle, and usually kick security tests to the end of that process as a cursory checkbox action item.

To keep things in perspective, when we look at scan frequency by application, we see that it’s still heavily weighted toward just a handful of scans per application. The median scan rate amongst our entire application portfolio under test is still just two. Plenty of organizations obviously still stick to what they’ve always done before.

FIGURE 41: SCAN RATES



Source: Veracode SOSS Volume 9, n=33.3k Static Scans

However, we'll note that there are a significant number of customers that are scanning their applications six or more times per year. Nearly one in three applications are scanned at that rate now. The numbers have fluctuated up and down slightly since we began tracking this, but for the most part, this rate of scanning has been fairly steady.

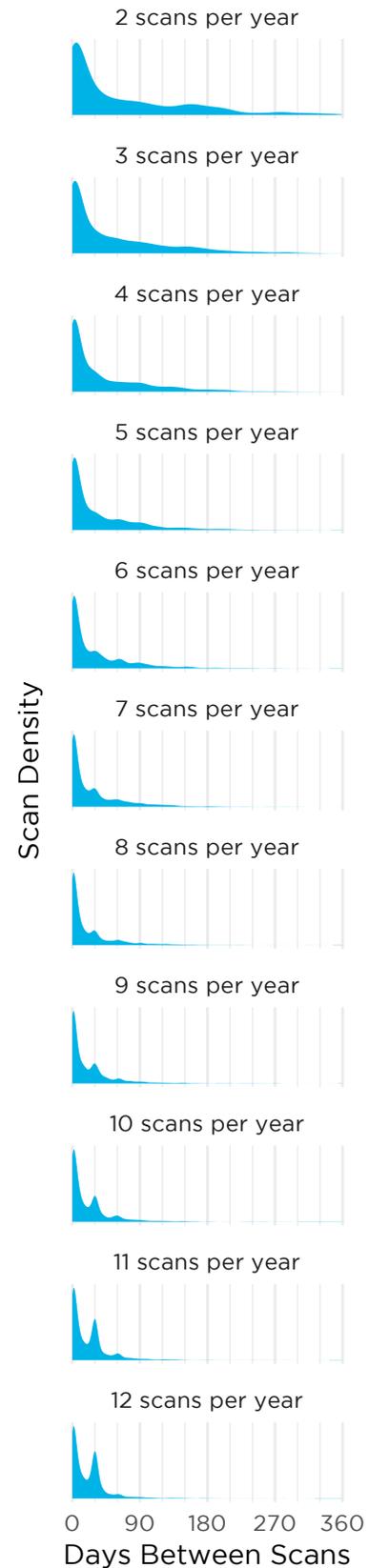
What this chart doesn't show is that there are some outliers in our customer base who have fully bought into the DevSecOps ethos. In some cases, we have customers who scan an application as many as 1,045 times per year. These DevSecOps unicorns are so intense in their rate of scan that they skew the average scan rate considerably. Whereas the median number of scans per year is two, the mean is more than seven scans annually.

In the past, we theorized that the number of scans completed were distributed fairly evenly throughout the course of the year. We assumed 12 annual scans probably indicated monthly scans, six scans indicated every-other-month tests and four scans indicated quarterly checks. This year, we decided to question those assumptions, and we're glad we did.

Interestingly, what we found is that a higher number of scans doesn't necessarily equate to a more frequent, regular cadence to security testing. Instead, when we looked at the distribution of scans, we found the most frequent occurrence of a rescan was just a day after the previous scan. Second to that was a rescan one week later. And the third most common pattern was a rescan three days after the previous scan.

When we looked at scan distribution based on the number of scans done per year, this consistently played out such that scans were typically conducted within only a few days or weeks of one another. As we got up to nine or more scans per year, we started to see an increase of rescans at 30-day intervals.

FIGURE 42: SCAN DISTRIBUTION



Source: Veracode SOSS Volume 9

When an application is scanned only two or three times in a year, and those scans are mostly done successively within a few days of one another, an obvious pattern emerges there. Clearly, many of these development teams are undergoing a process of doing their security checks, fixing the problems their organization's policies dictate, and then quickly moving on. This is same-old, same-old behavior.

But as we delve into scan distributions of organizations scanning six or more times a year, we see more rescans at weekly and monthly intervals, too. This spread could potentially be indicating sprint-based development practices that are popular among DevOps teams who frequently adhere to Agile and Scrum methods. Sprint development typically has teams working on a limited scope of work that's time-boxed, typically, into two-week- or month-long sprint cycles.

The data could be indicating trends where DevSecOps teams are working intensely on a particular application or app feature for one, two, or three focused sprint cycles, and wrapping up security scans within that work. In this case, it would make sense to see a number of scans popping off within a few days or a week or two of one another. The question is, are these security-focused sprints that are done so that a team can essentially ignore security for the rest of the year? Or are they feature-focused sprints that have security wrapped up into them? It's a difficult question to ask, but one which bears reflection.

We're still seeing some same-old, same-old behavior. When an application is scanned only two or three times in a year, and those scans are mostly done within a few days of one another, development teams are likely only fixing based on organization policy and then quickly moving on.

DevSecOps Increases Fix Velocity

Whatever the reason for the cadence of scanning, one thing is certain. Our data shows that there is a very strong correlation between how many times a year an organization scans and how quickly they address their vulnerabilities.

As we explained above, our working hypothesis is that a greater frequency of scans per year indicates a higher likelihood of DevSecOps adherence. Whether they officially call what they do 'DevOps,' 'Agile,' or something else entirely, we can show that the teams that are scanning more often are making incremental improvements every time they test.

This does amazing things for fix velocity.

FIGURE 43: FIX VELOCITY BASED ON SCAN FREQUENCY



Source: Veracode SOSS Volume 9

As you can see above, every jump in annual scan rates sees a commensurate step up in the speed of flaw fixes. Once organizations reach the point of 300 or more scans per year — the true territory of DevSecOps unicorns — the fix velocity goes into overdrive.

If we flip the discussion around and discuss flaw persistence intervals, we get greater visibility into how the frequency of scanning corresponds numerically to flaw persistence.

FIGURE 44: EFFECT OF SCAN FREQUENCY ON FLAW PERSISTENCE INTERVALS

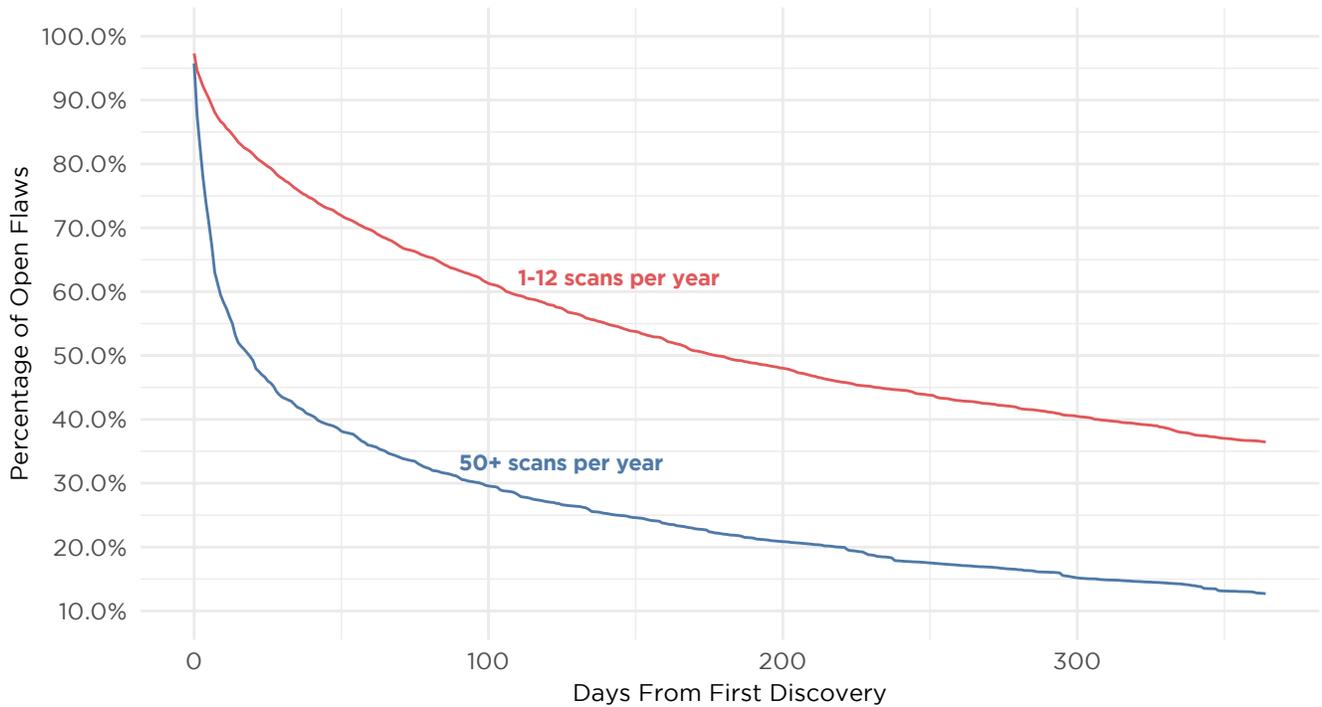


Source: Veracode SOSS Volume 9

If we look at flaw persistence intervals for those organizations that only scan a couple of times per year, we can see that it takes far longer than average to get around to making it to any one of the first three quartiles. When apps are tested fewer than three times a year, flaws persist more than 3.5x longer than when organization can bump that up to seven to 12 scans annually. At that rate of scan, flaw persistence intervals tend to track very closely to the average. Organizations really start to take a bite out of risk when they increase frequency beyond that. Each step up in scan rate results in shorter and shorter flaw persistence intervals. Once organizations are scanning more than 300 times per year, they're able to shorten flaw persistence 11.5x across the intervals compared to applications that are only scanned one to three times per year.

If we look at a simplified view of the flaw persistence analysis curves, the delta is imminently clear between those flaws that are rescanned 12 or fewer times per year and those that are checked on more than 50 times a year.

FIGURE 45: EFFECT OF SCAN FREQUENCY ON FLAW PERSISTENCE ANALYSIS



Source: Veracode SOSS Volume 9

It's important to note that this data may not necessarily be causal. And we admit that in some instances, more frequent scanning could just be detecting closures more quickly.

However, the correlation is strong enough to offer security professionals and developers alike some concrete evidence for why they should be embedding more frequent security checks into their SDLC.

We believe strongly that the same incremental processes and automation that DevSecOps teams put in place to make it easier to scan more frequently also lend themselves to faster remediation.

The data above offers some of the first ever statistical evidence to prove that out.

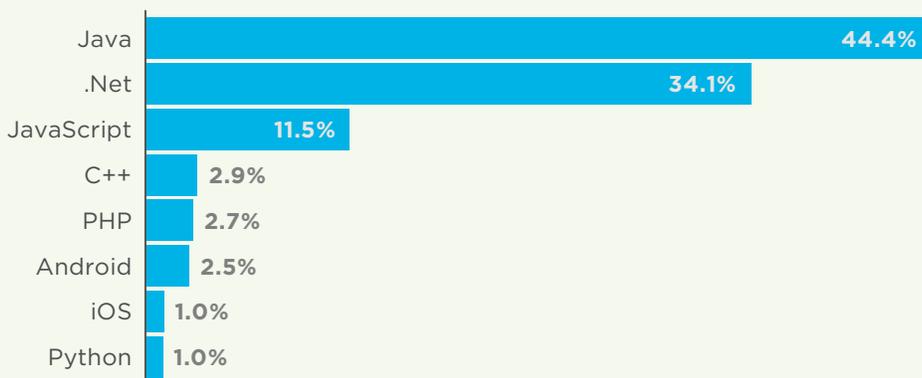
Development Trends and Risk



LANGUAGE AND COMPONENT USE

Most of the trends described in *SOSS Vol. 9* are seen through the lens of just a few major languages. Applications tested on the Veracode platform were most heavily weighted toward Java and .NET, with a healthy smattering of JavaScript. Beyond that, there were a number of other languages represented, but not in nearly the same numbers. This distribution likely reflects the development environments of the kinds of enterprises that Veracode caters to, not necessarily the development world as a whole.

FIGURE 46: LANGUAGE PREVALENCE

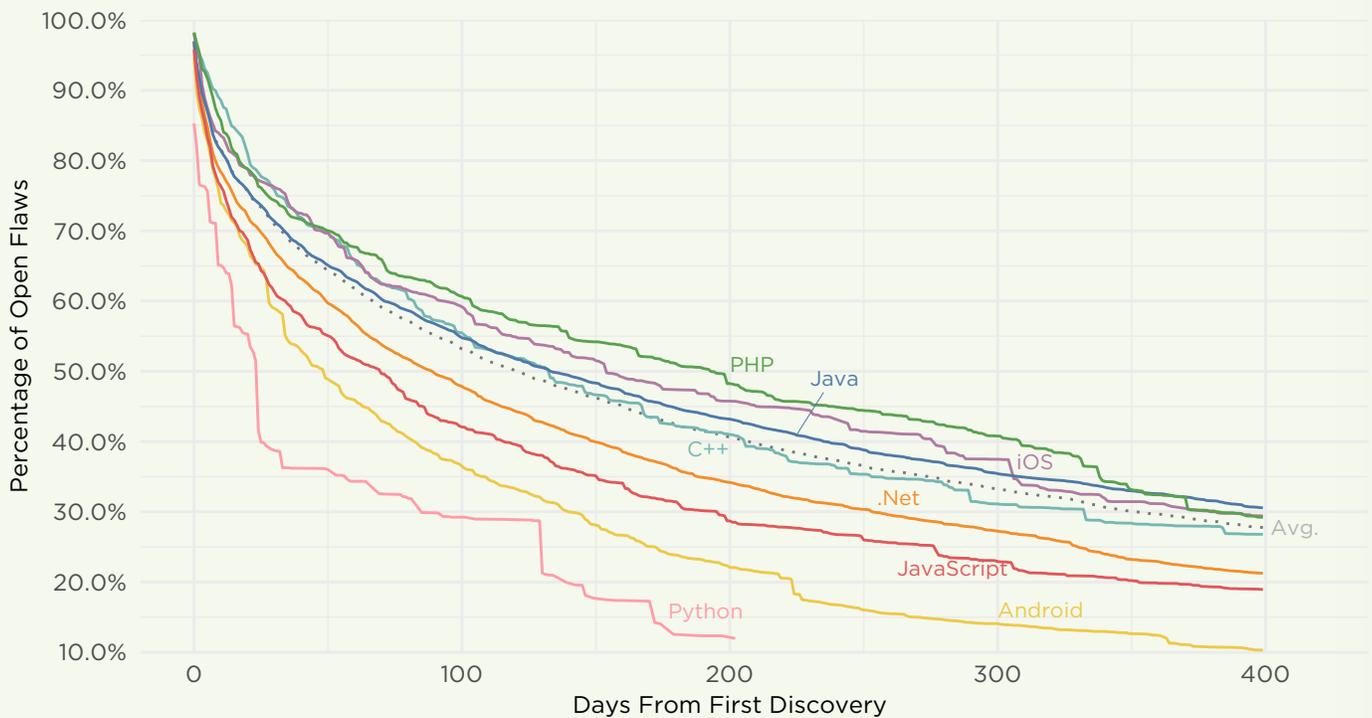


Percent of Applications

Source: Veracode SOSS Volume 9, n=32k

In examining flaw persistence analysis curves for the various languages represented in our mix, we see that .NET most closely follows the flaw persistence patterns of the average application. Developers were able to address flaws the quickest within software written in Javascript, Android, and Python. Meanwhile, they struggled more with longer open flaw windows in Java, iOS, and PHP.

FIGURE 47: LANGUAGE FLAW PERSISTENCE ANALYSIS



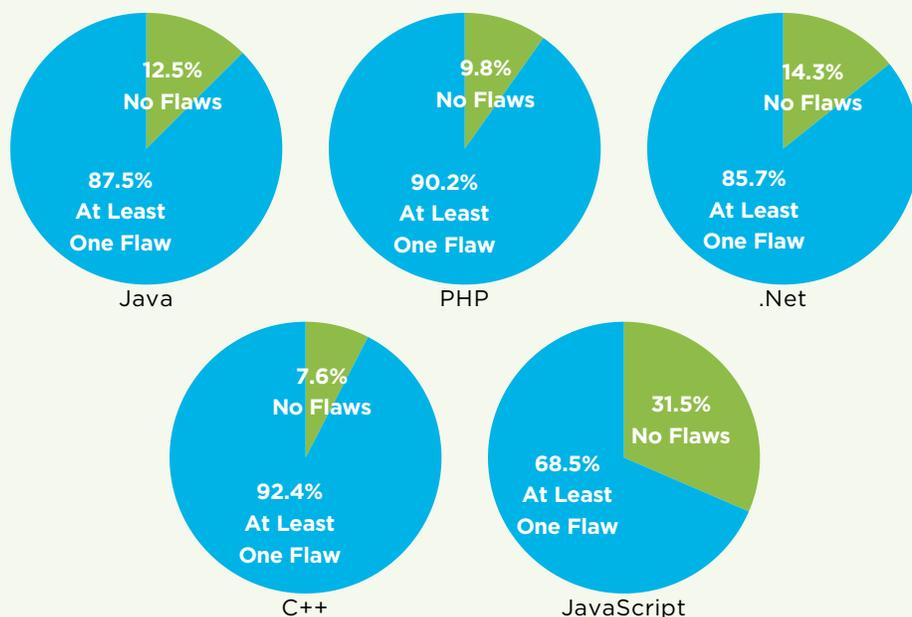
Source: Veracode SOSS Volume 9



SOFTWARE COMPOSITION ANALYSIS

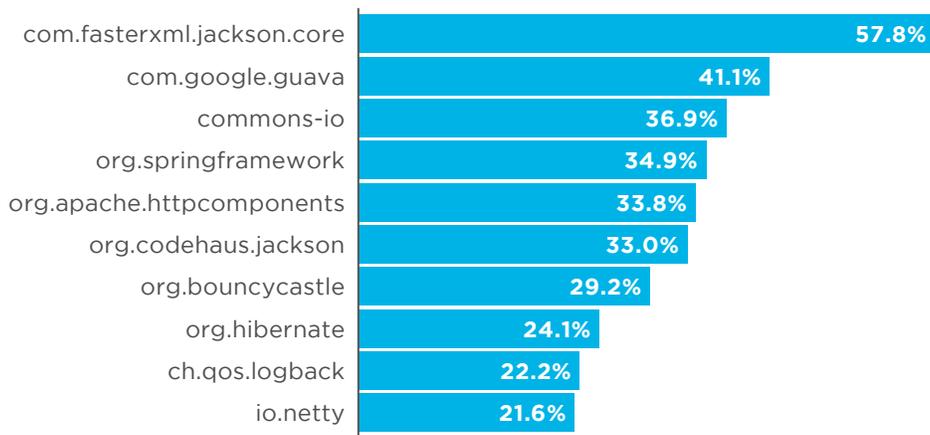
For several years now, we've drawn attention to the fact that vulnerable open source software components run rampant within most software. That trend continues. Last year, about 88% of Java applications had at least one vulnerability in a component; this year, that figure dipped down marginally to 87.5%.

FIGURE 48:
AT LEAST ONE FLAW IN A COMPONENT, BY LANGUAGE



Within typical languages, the majority of component flaws present themselves within flexible libraries and frameworks, which developers use in countless ways.

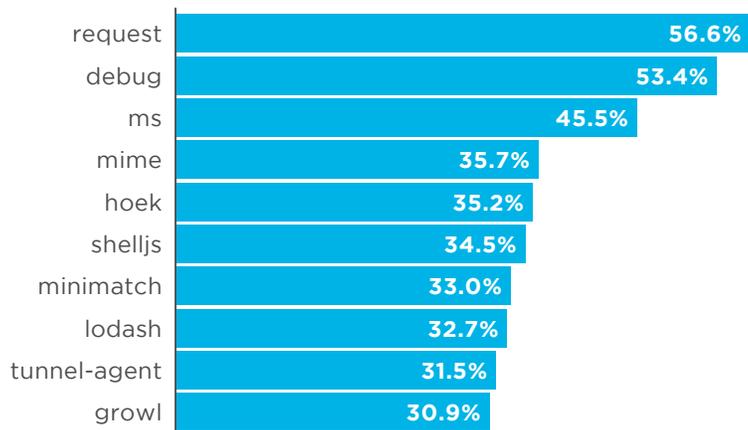
FIGURE 49: MOST COMMON JAVA COMPONENT FLAWS



Percent of Java Applications

Source: Veracode SOSS Volume 9, n=9,929

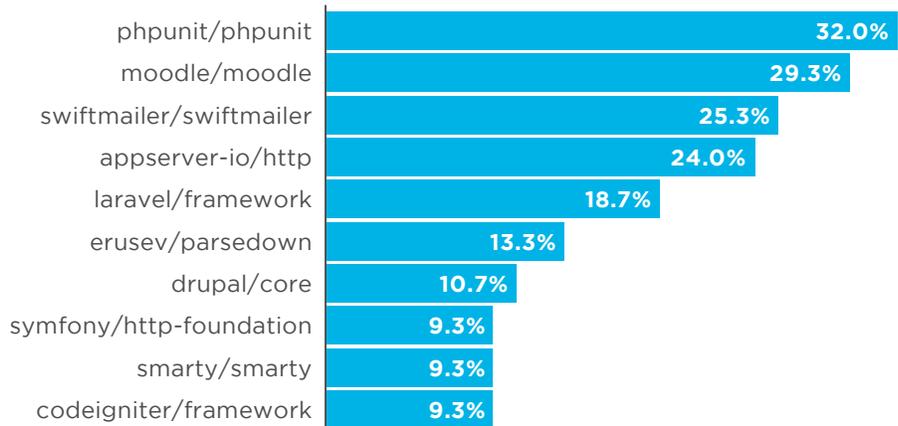
FIGURE 50: MOST COMMON JAVASCRIPT COMPONENT FLAWS



Percent of JavaScript Applications

Source: Veracode SOSS Volume 9, n=5,725

FIGURE 51: MOST COMMON PHP COMPONENT FLAWS



Percent of PHP Applications

Source: Veracode SOSS Volume 9, n=75

The crucial thing to keep in mind about vulnerable components is that it's not just important to know when a component contains a flaw, but whether that component is used in such a way that the flaw is easily exploitable. Data compiled from customer use of our SourceClear solution shows that at least nine times out of 10, developers are not necessarily using a vulnerable library in a vulnerable way.

By understanding not just the status of the component, but whether or not a vulnerable method is being called, organizations can pinpoint their component risk and prioritize fixes based on the riskiest uses of components. The charts above do not take vulnerable methods into account, which we believe bears future exploration in future reports.

Application Risk by Industry

Breaking out the application risk data by vertical offers security staff and developers in key industries strong benchmarks for comparing their AppSec performance to industry peers. The data also offers indications about which industries are making faster headway in improving their AppSec practices.

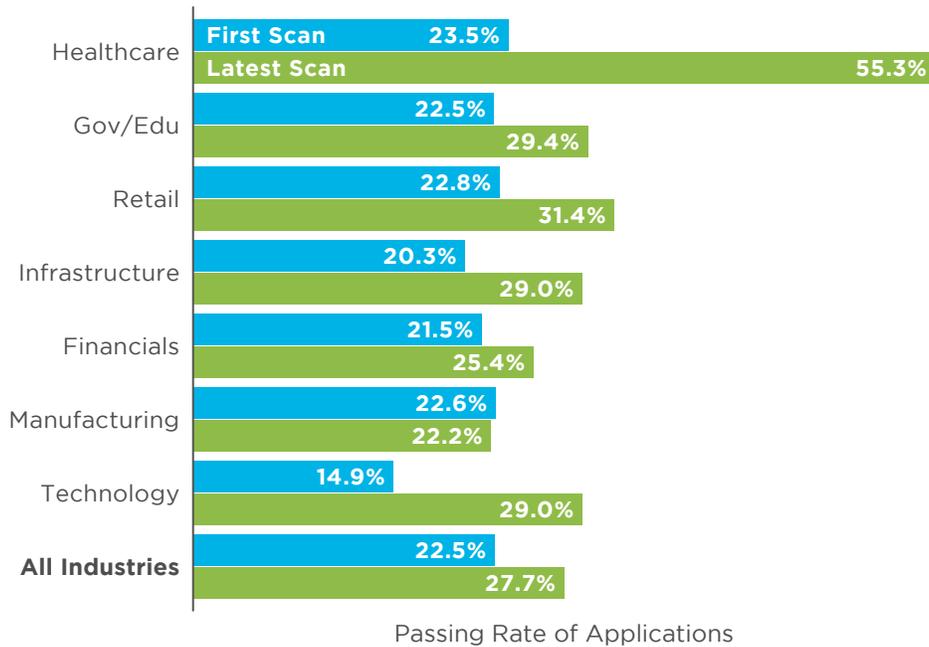
Industry Overview

This year's industry breakdowns show some interesting new trends, with nearly every industry making gains on application risk compared to last year's metric. Some specific verticals like Healthcare, Government and Education, and Retail made particularly huge strides, with those industries occupying the top three slots for OWASP pass rates on latest scan.

In looking at OWASP first scan/latest scan pairs, we see that all of the major industries outperform the overall rates by some margin in both metrics. To understand how that works, it's important to note that applications under test in these major industries only made up a percentage of all applications under test. About 15% of apps on first scan and 16% of apps in latest scan were done by organizations in other industries. Based on these numbers, those in the 'other' bucket are clearly underperforming compared to established verticals when it comes to passing the OWASP litmus test.



FIGURE 52: OWASP PASS RATES COMPARISON



Source: Veracode SOSS Volume 9

As readers compare improvement between first scan and latest scan pass results, we offer a caution. Some industries, like Healthcare, Retail, and Technology, saw an outsized jump in pass rates — but we need to keep in mind the number of applications tested in each round. There was a drastic drop in the number of applications retested by these organizations in all three industries.

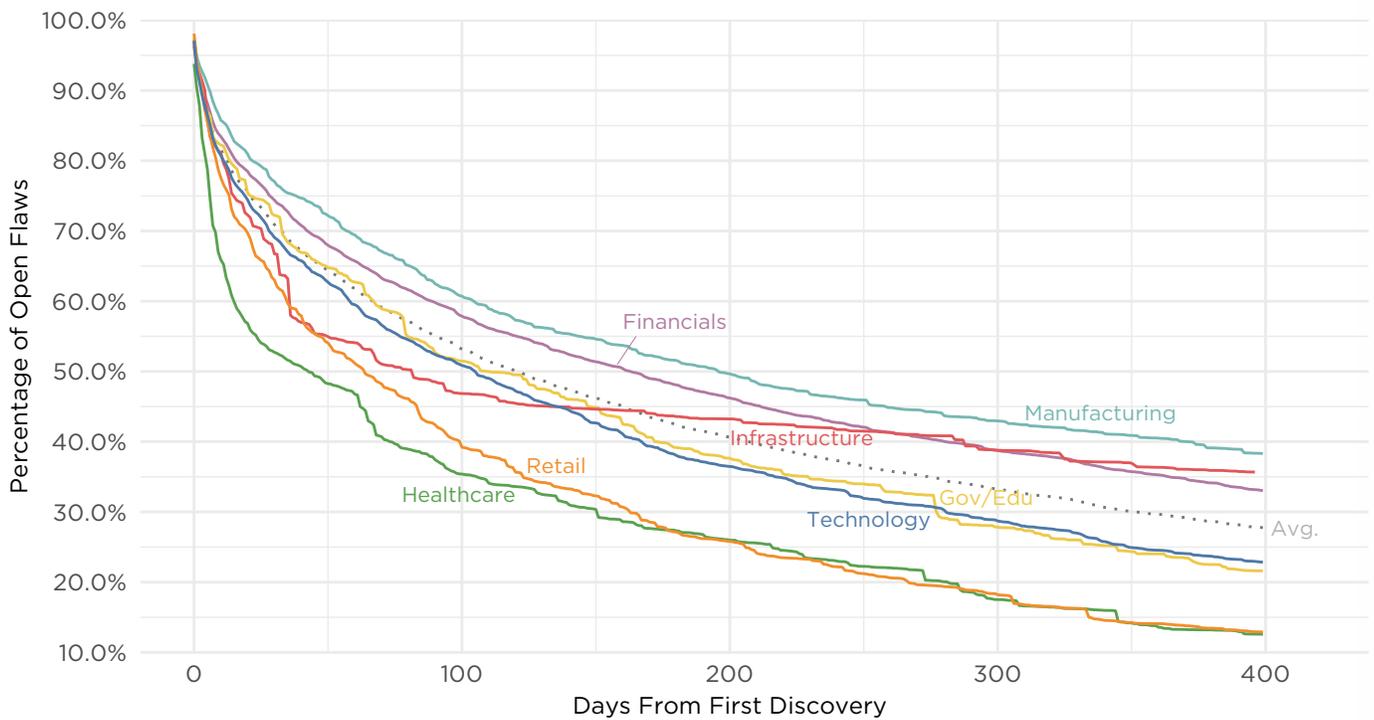
We have a couple of theories to explain this data. Our leading theory could be that organizations in these industries designate certain apps as important, and those are the ones they rescan, repeatedly, until they meet policy. These are highly regulated industries, so this behavior could indicate they're still doing a lot of testing for the auditors. The applications that aren't getting retested are the ones deemed less business critical, so they get scanned once for compliance and then ignored. That last rescan is simply a formality to get the passing checkmark for auditors, and everything else only tested one time is left by the wayside. Looking at it with this lens, you see that the first scan/latest scan improvements for industries like Government and Education, and Manufacturing are more likely to accurately reflect improvements to their entire portfolio over the course of the year. In these cases, the number of apps tested are nearly the same in both the first and latest scan.

In the “Focus on Fix” section of this report, we started the discussion of what flaw persistence intervals look like broken down by industry. Even with the caveat above about single scanning, when we look at flaw persistence among all discovered flaws, we see that Healthcare and Retail are still reducing their risk the fastest.



A comparative overview of flaw persistence analysis offers an at-a-glance view of how long each industry is letting risk linger relative to other verticals.

FIGURE 53: INDUSTRY OVERVIEW FLAW PERSISTENCE ANALYSIS



Source: Veracode SOSS Volume 9

Finally, the prevalence of the types of vulnerabilities plaguing organizations tend to vary slightly industry-by-industry – but they generally track with overall stats. This matrix compares the incidence of different vulnerability categories by industry.

FIGURE 54: INDUSTRY TOP VULNERABILITY CATEGORIES OVERVIEW

| | Overall | Financials | Gov/Edu | Healthcare | Infrastructure | Manufacturing | Retail | Tech |
|-------------------------------|---------|------------|---------|------------|----------------|---------------|--------|-------|
| API Abuse | 12.3% | 11.4% | 10.7% | 9.4% | 6.5% | 11.7% | 9.8% | 16.5% |
| Authentication Issues | 3.3% | 2.6% | 0.5% | 1.9% | 5.2% | 4.7% | 2.8% | 4.1% |
| Authorization Issues | 3.5% | 2.8% | 1.7% | 2.9% | 2.5% | 3.4% | 2.7% | 4.8% |
| Buffer Management Errors | 3.5% | 1.3% | 0.2% | 3.9% | 0.7% | 3.9% | 1.4% | 9.1% |
| Buffer Overflow | 3.0% | 0.7% | 0.3% | 4.2% | 0.2% | 3.4% | 1.2% | 8.9% |
| Code Injection | 7.9% | 6.6% | 4.8% | 6.1% | 8.2% | 6.2% | 9.2% | 10.3% |
| Code Quality | 63.1% | 62.7% | 59.1% | 59.4% | 63.1% | 61.6% | 65.0% | 64.2% |
| Command or Argument Injection | 13.7% | 11.0% | 6.7% | 14.2% | 5.7% | 12.3% | 13.2% | 19.4% |
| Credentials Management | 43.0% | 40.2% | 34.8% | 39.8% | 38.2% | 36.9% | 43.4% | 50.1% |
| CRLF Injection | 59.5% | 62.5% | 50.5% | 50.3% | 59.1% | 55.6% | 59.1% | 56.6% |
| Cross-Site Scripting (XSS) | 48.6% | 48.9% | 58.8% | 44.5% | 32.7% | 44.0% | 47.2% | 48.5% |
| Cryptographic Issues | 63.7% | 61.0% | 41.7% | 62.1% | 47.9% | 59.3% | 64.3% | 70.1% |
| Dangerous Functions | 2.1% | 0.5% | | 2.9% | | 1.8% | 0.5% | 6.4% |
| Directory Traversal | 48.0% | 43.5% | 37.9% | 47.2% | 32.9% | 49.6% | 51.0% | 55.5% |
| Encapsulation | 19.9% | 21.0% | 17.3% | 15.0% | 12.5% | 17.9% | 18.9% | 20.5% |
| Error Handling | 7.7% | 4.5% | 1.9% | 10.4% | 2.7% | 9.1% | 4.9% | 14.5% |
| Information Leakage | 66.9% | 67.3% | 64.5% | 63.8% | 60.8% | 61.7% | 66.1% | 67.8% |
| Insecure Dependencies | 2.5% | 2.3% | 1.4% | 2.9% | 2.0% | 2.7% | 2.6% | 2.7% |
| Insufficient Input Validation | 46.6% | 48.6% | 50.4% | 45.8% | 45.4% | 38.7% | 46.3% | 44.1% |
| Numeric Errors | 3.1% | 0.7% | 0.2% | 4.2% | 0.2% | 3.6% | 1.1% | 8.9% |
| Potential Backdoor | 9.0% | 5.8% | 4.8% | 12.6% | 4.0% | 8.0% | 8.0% | 16.4% |
| Race Conditions | 8.1% | 7.7% | 6.1% | 5.0% | 3.0% | 6.2% | 6.2% | 11.7% |
| Session Fixation | 9.5% | 10.6% | 8.7% | 7.1% | 4.7% | 9.1% | 9.9% | 8.4% |
| SQL Injection | 27.5% | 25.7% | 24.0% | 31.9% | 13.7% | 24.0% | 31.5% | 31.9% |
| Time and State | 19.4% | 17.2% | 14.5% | 19.3% | 20.9% | 17.8% | 22.2% | 24.1% |
| Untrusted Initialization | 10.9% | 8.6% | 5.4% | 16.8% | 8.2% | 8.2% | 12.9% | 15.5% |
| Untrusted Search Path | 6.9% | 4.8% | 2.2% | 9.7% | 5.2% | 6.5% | 6.3% | 11.3% |

Source: Veracode SOSS Volume 9

Industry Snapshots

Financials

Undeniably, the largest population of applications under test come from the Financial vertical. While financial organizations tend to have the reputation of having some of the most mature overall cybersecurity practices, our data shows they struggle like the rest of organizations to stay on top of application security.

The industry ranked second to last in the major verticals for latest scan OWASP pass rate, and based on the flaw persistence analysis chart, it is leaving flaws to linger longer than other industries.

LATEST SCAN PASS RANK

| 2018 | 2017 |
|------|------|
| 6 | 5 |

Source of all charts:
Veracode SOSS Volume 9

FIGURE 55: FINANCIALS LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 56: FINANCIALS TOP VULNERABILITY CATEGORIES

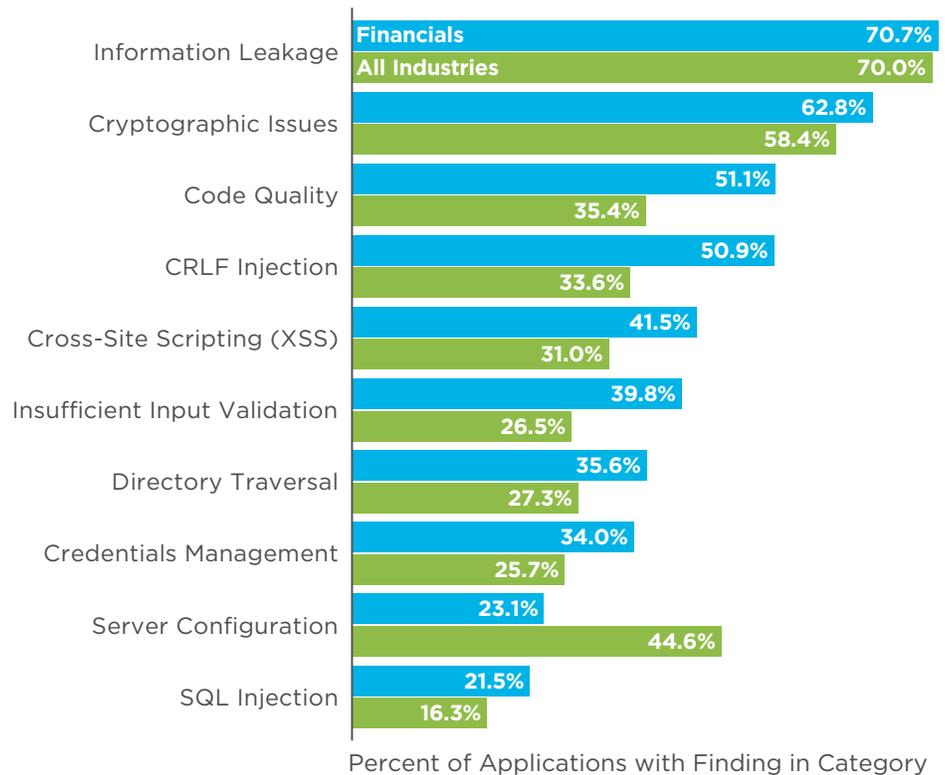
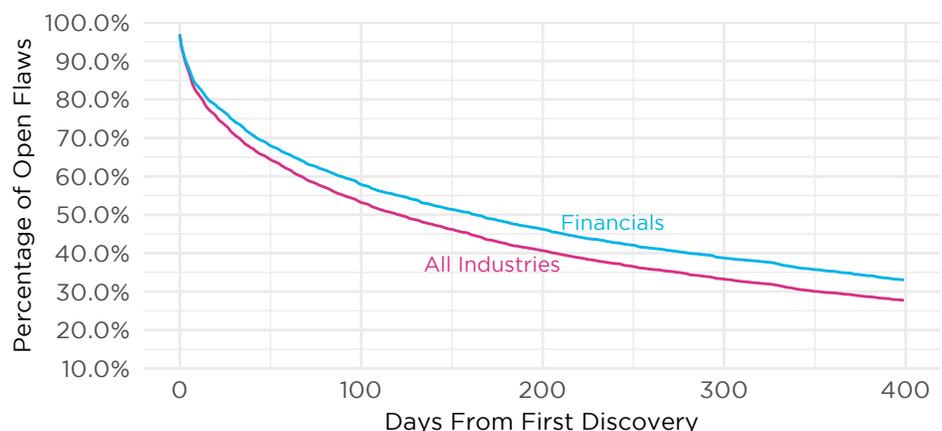


FIGURE 57: FINANCIALS INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Government and Education

This year's data holds a lot of good news from the Government and Education sector, which performed significantly better than in volume 8 of this report. Last year, the industry was dead last in latest scan OWASP pass rank. This year, it came in second only to Healthcare. Its OWASP pass rate is about 20 percentage points higher this year, and the remarkable thing about this is that organizations in Government scan just as many apps in latest scan as they do in first scan.

In examining flaw persistence, the analysis curve shows that while these organizations are slower than usual out of the gate, they pick up speed with resolving vulnerabilities as they dig into the second half of remaining flaws.

| LATEST SCAN PASS RANK | |
|-----------------------|------|
| 2018 | 2017 |
| 5 | 7 |

Source of all charts: Veracode SOSS Volume 9

FIGURE 58: GOVERNMENT AND EDUCATION LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 59: GOVERNMENT AND EDUCATION TOP VULNERABILITY CATEGORIES

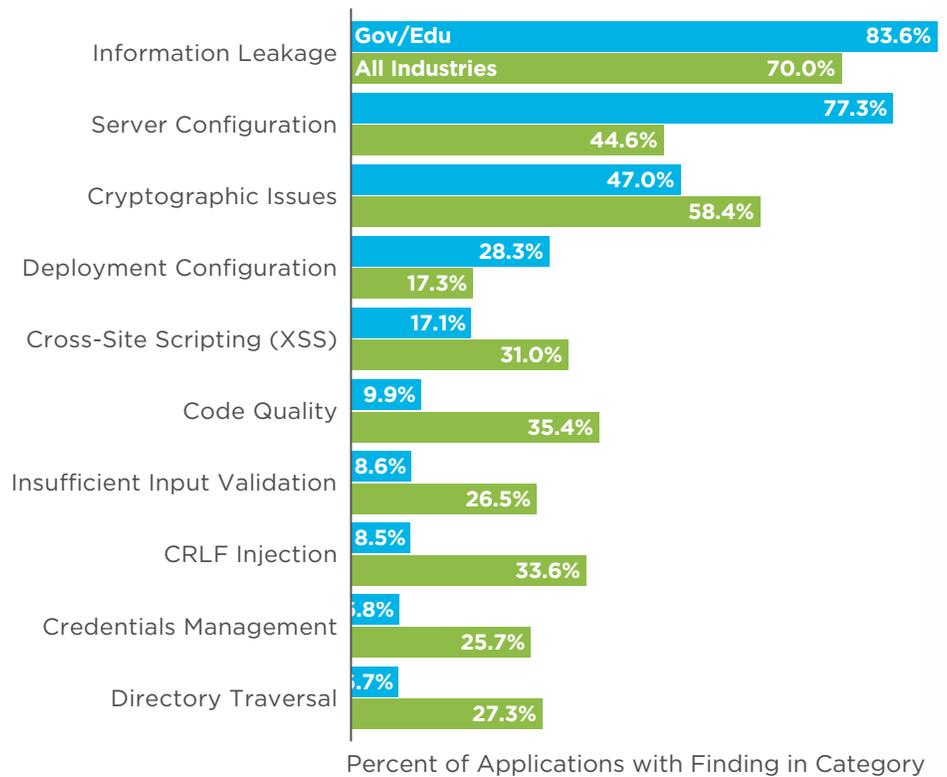
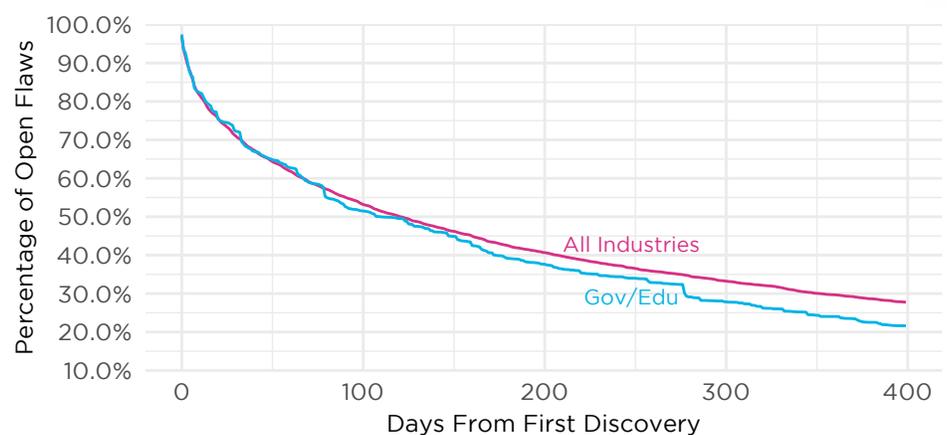


FIGURE 60: GOVERNMENT AND EDUCATION INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Healthcare

The highly regulated healthcare industry got high marks in many of this year's SOSS metrics. Organizations in this sector had the highest latest scan OWASP pass rates of all verticals, though we will reiterate that the population of apps scanned was significantly lower than for first scan results. This indicates that healthcare organizations could be leaving some risk on the table with many applications scanned only a single time and subsequently ignored.

Nevertheless, flaw persistence analysis shows that when looking at all found vulnerabilities, this industry is statistically closing the window on app risk more quickly than any other sector.

LATEST SCAN PASS RANK

| 2018 | 2017 |
|------|------|
| 1 | 2 |

Source of all charts:
Veracode SOSS Volume 9

FIGURE 61: HEALTHCARE LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL

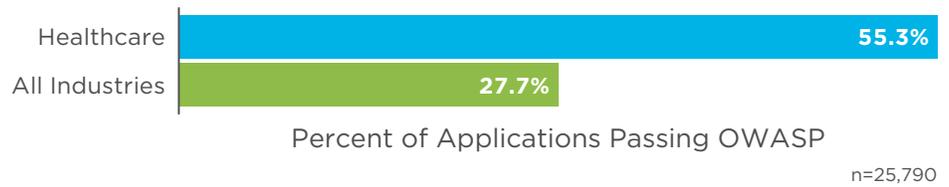


FIGURE 62: HEALTHCARE TOP VULNERABILITY CATEGORIES

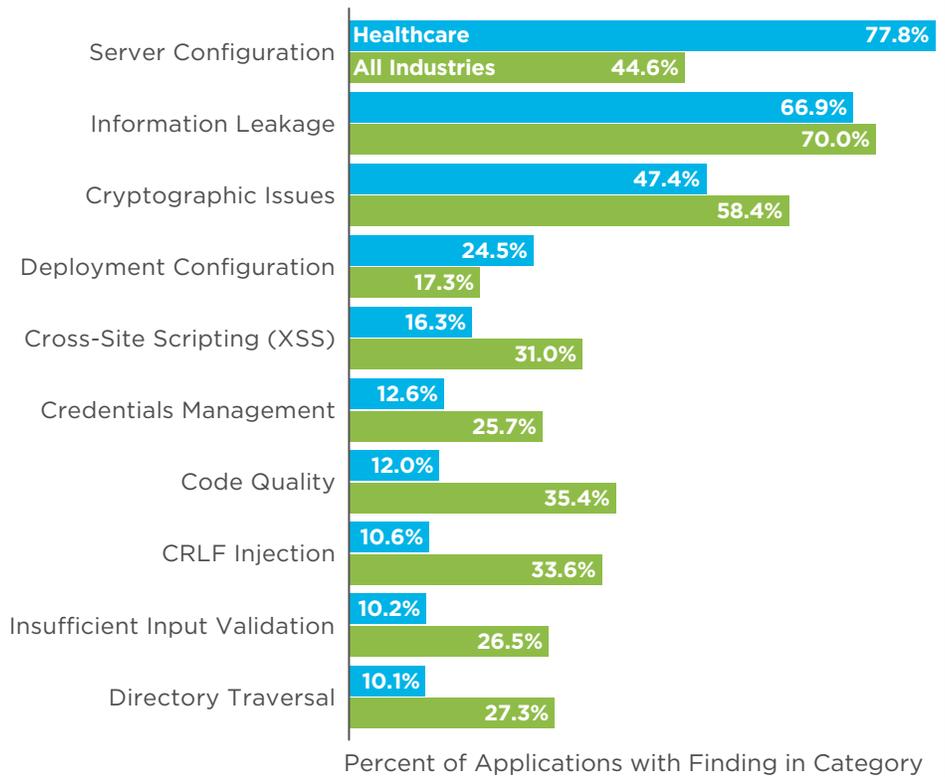
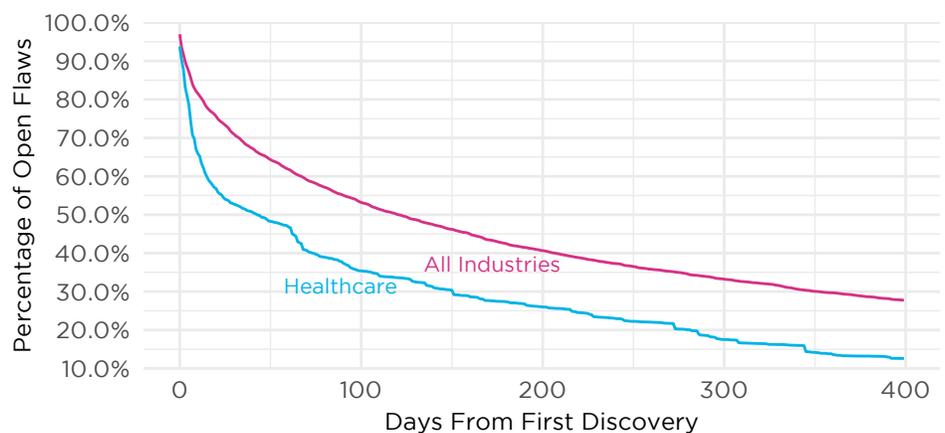


FIGURE 63: HEALTHCARE INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Infrastructure

Infrastructure organizations test the fewest number of applications compared to any other tracked vertical, despite the growing risk to their applications.

Infrastructure organizations ranked toward the bottom of the list when it comes to latest scan OWASP pass rates. The good news is that they still saw a bump in this metric, gaining about 6 percentage points over similar 2017 pass rates.

In examining flaw persistence, infrastructure jumped on the first half of their flaws very quickly relative to the average. But organizations in this sector struggled to take care of the last 50% in a timely manner. This likely indicates the unique challenges of the vertical, which is chock full of sensitive applications with low tolerance for downtime and stringent change management practices that may delay the deployment of code fixes.

LATEST SCAN PASS RANK

2018 2017

5

3

Source of all charts:
Veracode SOSS Volume 9

FIGURE 64: INFRASTRUCTURE LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 65: INFRASTRUCTURE TOP VULNERABILITY CATEGORIES

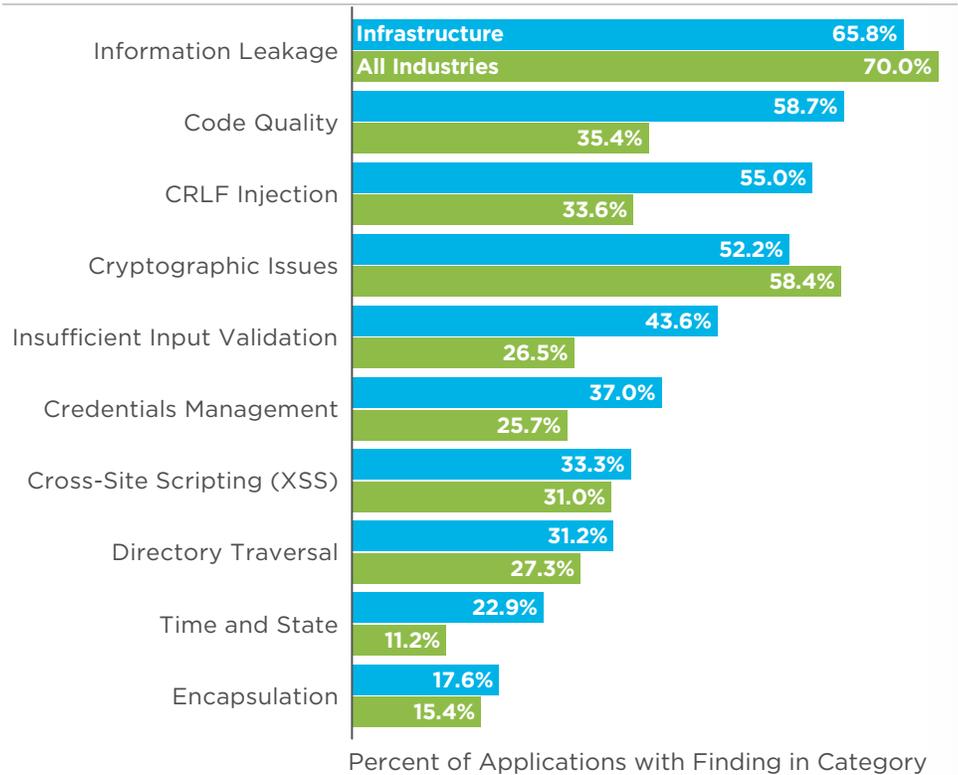
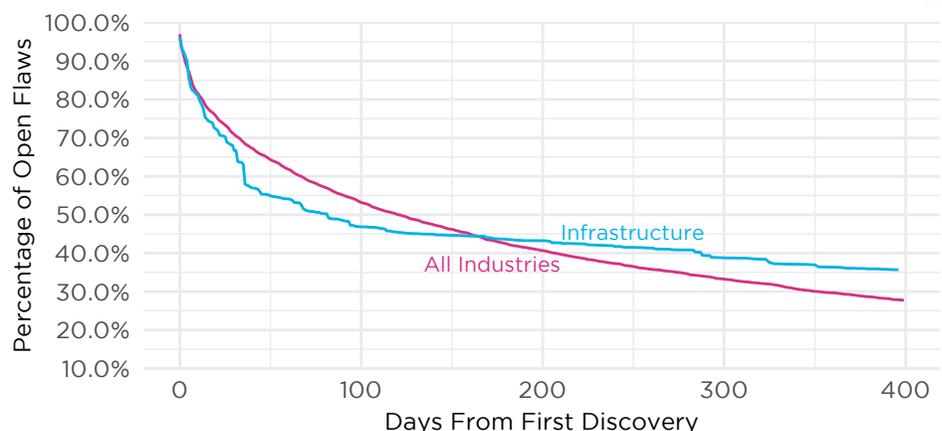


FIGURE 66: INFRASTRUCTURE INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Manufacturing

The manufacturing industry tumbled the farthest in the rankings for latest scan OWASP pass rates, dropping from first to last industrywide. But when we examined the actual percentages year-by-year we found that the sector had nearly the identical proportion of applications passing OWASP standards on latest scan this year compared to last year.

This indicates that even though manufacturing didn't lose ground on OWASP adherence, it isn't improving the way other industries did in the last year. When we look at flaw persistence, manufacturing clearly has a lot of work to do. It consistently left application risks to linger longer than any other industry.

One notable piece of data for this industry: it was the only one with lower latest scan OWASP pass rates than first scan. This could be an indication of more new applications under test for this industry this year.

LATEST SCAN PASS RANK

| 2018 | 2017 |
|------|------|
| 7 | 1 |

Source of all charts:
Veracode SOSS Volume 9

FIGURE 67: MANUFACTURING LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 68: MANUFACTURING TOP VULNERABILITY CATEGORIES

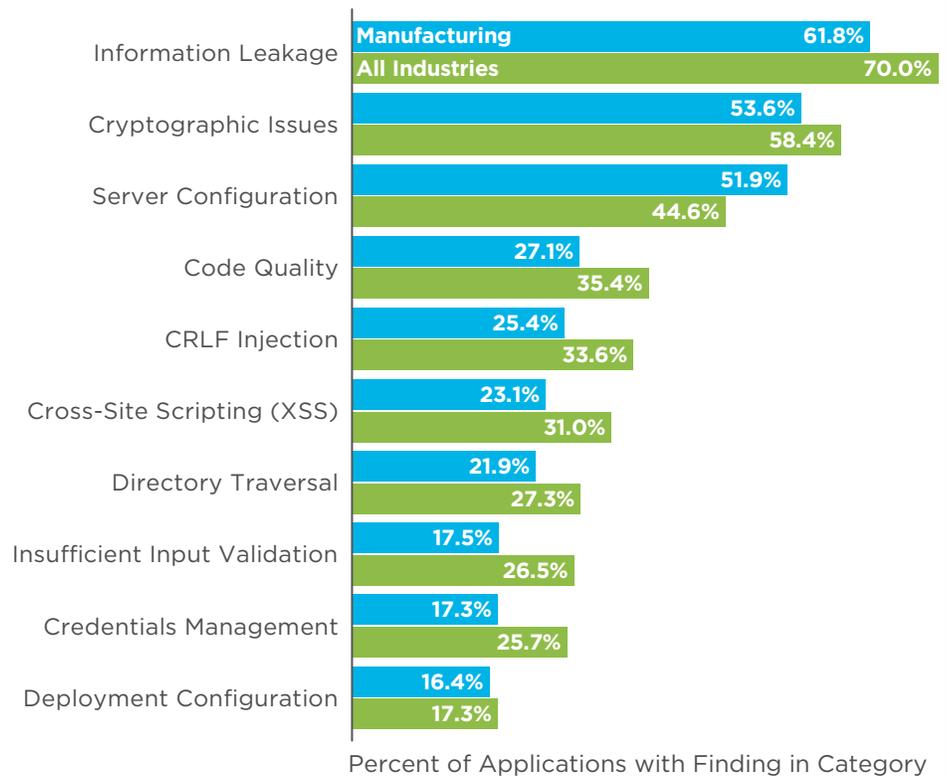
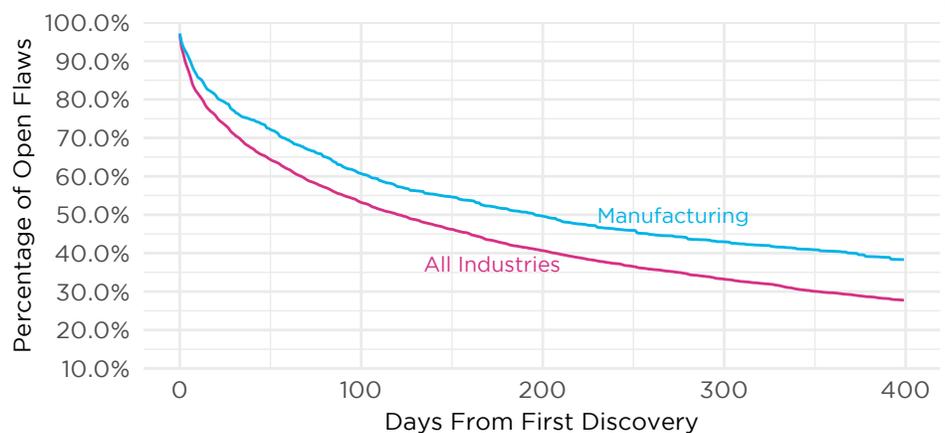


FIGURE 69: MANUFACTURING INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Retail

Retail offered another bright spot in this mix of industries. Its latest scan OWASP pass rates improved decently by about 12 percentage points in the last year, and it edged from fourth to third place in this regard.

It is also notable how much less time this vertical leaves its flaws open compared to almost all other sectors. The flaw persistence analysis curve for the Retail category shows that it's only second to Healthcare in its speed of shutting down flaws.

LATEST SCAN PASS RANK

2018

2017

3

4

FIGURE 70: RETAIL LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 71: RETAIL TOP VULNERABILITY CATEGORIES

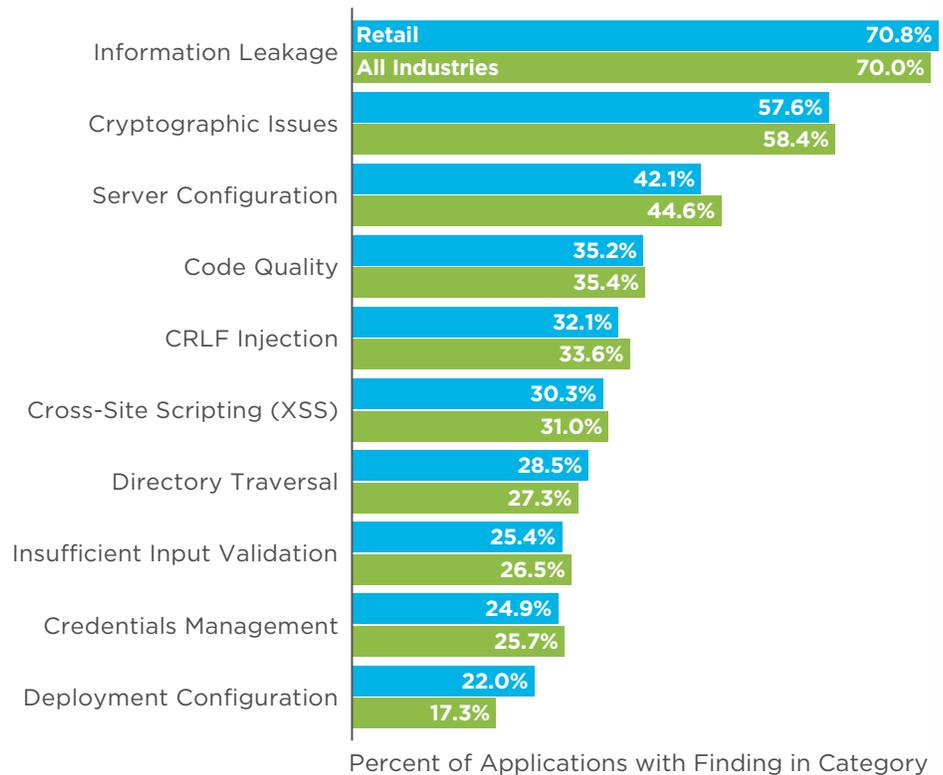
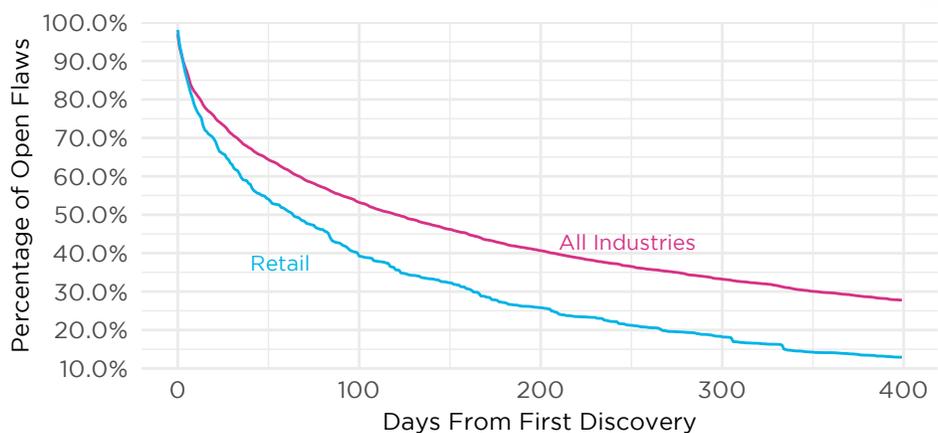


FIGURE 72: RETAIL INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



Source of all charts: Veracode SOSS Volume 9

Technology

Technology is the second-most prolific industry in terms of the volume of apps tested. Only financial organizations test more applications on the Veracode platform. As a group, tech companies occupy middle-of-the-pack status for most performance indicators.

It came in fourth for latest scan OWASP pass rates, and its pass rate this year was a healthy 11 percentage points higher than last year's results. For flaw persistence, technology firms have a curve that sits right between other comparable industry curves. In examining the flaw persistence analysis curve for tech firms compared to the overall curve, we can see that the industry leaves flaws present for less time than the typical firm.

LATEST SCAN PASS RANK

2018

4

2017

6

Source of all charts:
Veracode SOSS Volume 9

FIGURE 73: TECHNOLOGY LATEST SCAN OWASP PASS RATE COMPARED TO OVERALL



FIGURE 74: TECHNOLOGY TOP VULNERABILITY CATEGORIES

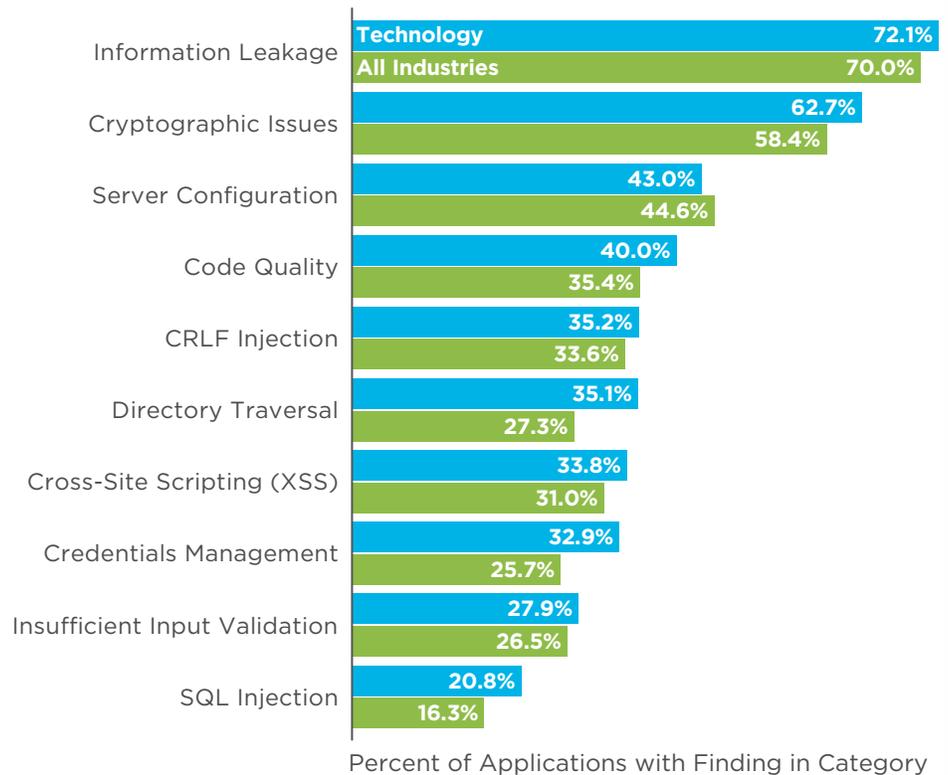
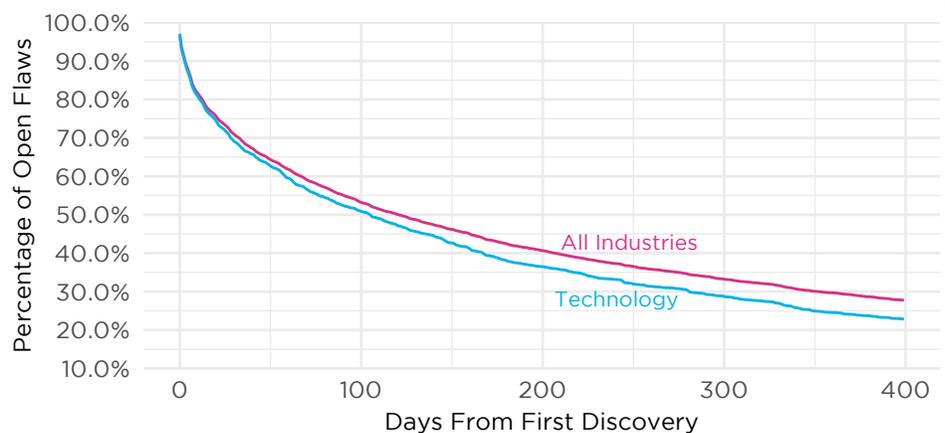


FIGURE 75: TECHNOLOGY INDIVIDUAL VERTICAL FLAW PERSISTENCE ANALYSIS



KEY TAKEAWAYS

As with so many of the Veracode SOSS reports of the past, SOSS Vol. 9 is full of good and bad news about enterprise progress on advancing application security. The data offers many signs of encouragement that organizations are incrementally moving the needle on application security. At the same time, these positive indicators are balanced by other evidence showing there's still plenty of work to be done to shore up application risk.

Here are the key lessons we believe that security professionals, developers, and business executives should take from the data.

Fix Velocity Matters

The speed at which organizations fix flaws they discover in their code directly mirrors the level of risk incurred by applications. The faster organizations close vulnerabilities, the less risk software poses over time.

Consider All Dimensions of Risk

The sheer volume of open flaws within enterprise applications is too staggering to tackle at once. Which means that organizations need to find effective ways to prioritize which flaws they fix first. While many organizations are doing a good job prioritizing by flaw severity, data this year shows that they're not effectively considering other risk factors such as the criticality of the application or exploitability of flaws.

DevSecOps Works

SOSS Vol. 9 offers some of the most dramatic and concrete evidence to date on the positive effect DevSecOps practices have on the state of software security. The data showed consistently that the more an organization scans per year, the faster security fixes are made. The frequent, incremental changes brought forth by DevSecOps makes it possible for these teams to fix flaws lightning fast compared to the traditional dev team.

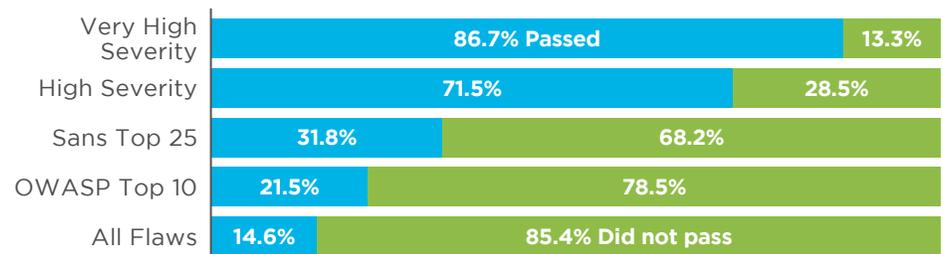
Components Still Thwart Enterprises

Enterprises still struggle with the occurrence of vulnerable open source components within their software. As organizations tackle bug-ridden components, they should consider not just the open flaws within libraries and frameworks, but also how those components are being used. Some component flaws may have mitigating factors if they're not being used in such a way that the flaw is exposed to exploit.

There's Still a Lot of Work to Do

The following is the overall pass rate of all scans compiled for SOSS Vol. 9. Clearly the industry still has a lot of work to do. The time to get started is now!

FIGURE 76: OVERALL PASS RATE



Percent of Applications

Source: Veracode SOSS Volume 9, n=456.4k scans

Methodology

About the dataset

Veracode methodology for data analysis uses statistics from a 12-month sample window. The data represents more than 700,000 application assessments submitted for analysis during the 12-month period from April 1, 2017 through March 31, 2018, except for the time-to-close flaws data. The data represents large and small companies, commercial software suppliers, open source projects, and software outsourcers. In most analyses, an application was counted only once, even if it was submitted multiple times as vulnerabilities were remediated and new versions uploaded.

The report contains findings about applications that were subjected to static analysis, dynamic analysis, software composition analysis, and/or manual penetration testing through Veracode's cloud-based platform. The report considers data that was provided by Veracode's customers (application portfolio information such as assurance level, industry, application origin) and information that was calculated or derived in the course of Veracode's analysis (application size, application compiler and platform, types of vulnerabilities, and Veracode Level — predefined security policies which are based on the NIST definitions of assurance levels).

A Note on Mass Closures

While preparing the data for our analysis, and exploring the flaw persistence visualizations (many of which made it into the report), we noticed several large single-day “drops” in the charts. While it's not strange for a scan to discover that dozens or even hundreds of findings have been fixed (50% of scans closed three or less findings, 75% closed less than 8), we did find it strange to see some applications closing thousands of findings in a single scan. Upon further exploration, we found many of these to be invalid: developers would scan entire filesystems, invalid branches or previous branches, and when they would rescan on the valid code, every finding not found again would be marked as “fixed.” These mistakes had a large effect: the top one-tenth of one-percent of the scans (0.1%) accounted for almost a quarter of all the closed findings. These “mass closure” events had a significant effect on exploring flaw persistence and the fix velocity, and were ultimately excluded from the persistence analysis.

Industry Verticals

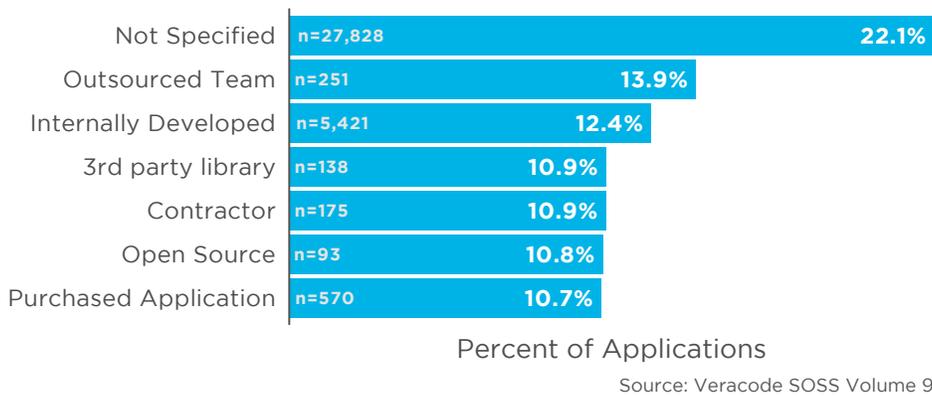
This report condenses information about applications coming from 38 different industry classifications into seven industry verticals, plus a bucket for “other.” The component industry classifications come from Data.com via Salesforce.com, but Veracode has created the industry verticals below to simplify the analysis. This year’s *State of Software Security* report adds a new industry vertical grouping for infrastructure, based on increased sample size in these industries (previously included in “other”), and due to increased attention to security in the component industries. In this year’s report, education organizations were added to the government industry vertical. A mapping of the component industries to industry verticals is provided below.

| Industry Vertical | Component industries as Defined in Data.com |
|---------------------------|--|
| Financial services | Banking, Finance, Insurance |
| Government | Government, Education |
| Healthcare | Healthcare, Pharmaceuticals |
| Infrastructure | Energy, Transportation, Utilities |
| Manufacturing | Manufacturing, Aerospace |
| Other | Biotechnology, Entertainment, Not for Profit, Apparel, Communications, Engineering, Media, Media & Entertainment, Food & Beverage, Machinery, Construction, Chemicals, Shipping, Business Services, Automotive & Transport, Beverages, Recreation, Real Estate, Membership Organizations, Environmental, Consumer Services, Not Specified, Other |
| Retail | Retail, Hospitality |
| Technology | Technology, Telecommunications, Electronics, Software, Security Products and Services, Consulting, Computer Hardware |

Data by App Purpose

Veracode collects optional metadata about the application purpose of software scanned by the platform — this data is provided voluntarily by customers. Since the data set was not as complete as most of our other information, we did not feel comfortable sharing findings in the main report but we still believe it is worth sharing. When customers did specify app purpose we found a fairly even distribution between different types of applications — including internally developed apps, those developed by an outsourced team or contractors, open source applications and commercial, off-the-shelf applications purchased by the testing organization.

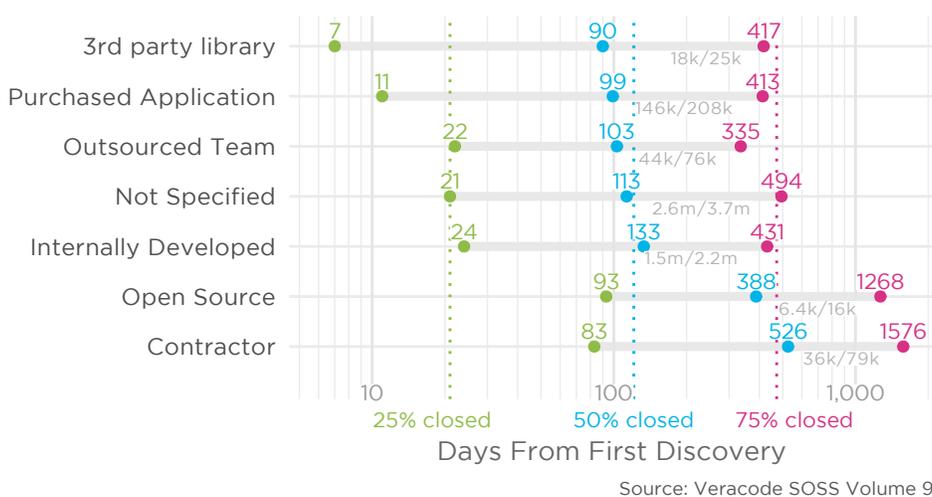
FIGURE 77: OVERALL PASSING BY PURPOSE



In examining the data, we found that there were some large differences in flaw persistence intervals between the different kinds of applications.

For example, third-party libraries were some of the fastest selections of code to be fixed, while overall open source software was among the slowest. Meantime, internally developed applications tracked almost dead-on with average flaw persistence intervals.

FIGURE 78: FLAW PERSISTENCE BY APP PURPOSE



VERACODE

STATE OF SOFTWARE SECURITY

VOLUME 9



Contact Us

Veracode can help
secure your applications

ABOUT VERACODE

Veracode 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 helps 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 more than 1,400 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 Forbes' 100 Most Valuable Brands. Learn more at www.veracode.com, on the **Veracode blog**, on **Twitter** and in the **Veracode Community**.

Copyright © 2018 Veracode, Inc. All rights reserved. All other brand names, product names, or trademarks belong to their respective holders.