Contrast SECURITY

2021 REPORT

2021 Application Security Observability Report

Understanding Application and API Risk by Connecting Real-world Vulnerabilities With Actual Attack Data

Table of contents

Table of contents

Table of contents

01 Foreword

01 | Foreword

Disruptive change in the marketplace has been the name of the game for some time, and this trend promises to continue in the months and years to come. Businesses were already embracing digital transformation before the COVID-19 pandemic, which rapidly accelerated digital adoption—with McKinsey indicating late last year that the share of digital and digitally enabled products leapfrogged an astounding seven years in the past year.[1]

Companies with the agility to evolve with current trends and quickly tap new revenue opportunities are best positioned to survive and thrive in the new economy. And more than ever before, software applications are a critical part of that agility. Yet, the application layer is an increasingly attractive target for cyber criminals, with high-profile software supply chain attacks on SolarWinds, Microsoft Exchange, and Kaseya making headlines in the past six months. Overall, 39% of data breaches in the past year have been the result of an application vulnerability.[2]

In this context, we are pleased to present Contrast Labs' second annual Application Security Observability Report. Our flagship research report covers the past 12 months of telemetry data from our different product offerings—Contrast Assess, Contrast Protect, and Contrast OSS. Contrast is the only player in the marketplace to present findings from vulnerability, attack, and open-source library data in a single report. This year, we added data on application makeup and the vulnerability escape rate (VER), both of which bring fascinating insights. In addition, we also added data on which potential routes are exercised within applications using data from our Route Intelligence functionality, released in March 2020.

Our goal with this report is to guide organizations as they strive to grow digital innovation while maintaining application security. Accordingly, we focus first on the nuts and bolts of how software is structured—the active and inactive code that makes up the software and how many routes are exercised. We then focus on custom code vulnerabilities and open-source libraries and frameworks. We then move to data about vulnerability remediation and security debt and to telemetry about application attacks.Finally, we update the Contrast RiskScore Index, which provides an easy-to-understand score that helps organizations better understand the risk of various vulnerability types.

Throughout these analyses, a few themes pop up repeatedly. First and foremost, effective application security is a matter of good prioritization of prevention and response efforts. This year, we find that only 6% of the code in a typical application is from active third-party library classes, while 20% is custom code. This means that 74% of the code in applications is third-party content that is never invoked by the application, and vulnerabilities in this code pose no risk to an organization and should be deprioritized.

A second recurring theme is the need for comprehensive observability into all aspects of application security throughout the software development life cycle (SDLC). This includes immediate feedback when a vulnerability is created, visibility into which open-source content is active, and which attacks are probes versus those that target existing vulnerabilities and hence could cause damage. Such observability enables organizations to deal with new vulnerabilities as they occur and to reduce their security debt over time.

This brings us to a third theme in the research: the need to resolve vulnerabilities quickly to ensure the timely delivery of secure software into production. Our research finds that Contrast customers deliver on this priority almost 29 times more quickly than customers of a well-known vendor of traditional application security tools, according to their research. And the Contrast platform has the added benefit of helping developers learn to introduce fewer vulnerabilities into their code, reducing their vulnerability escape rate (VER) from initial deployment to one year by 92%.

Our 2021 report provides a snapshot in time of where things stand with the applications developed by the Contrast customer community. The security observability provided by the Contrast Application Security Platform enables their development, security, and operations teams to work together to secure their software—and deliver to increasingly aggressive deadlines in a fast-changing marketplace.

Sincerely,

Jeff Williams
Cto and Co-Founder

David Lindner
Chief Information Security Officer

02 Executive Summary

02 | Executive Summary

Contrast Labs' second annual Application Security Observability Report takes a broad look at the state of application security at a critical moment in the economy—while digital transformation continues at an accelerated pace, businesses reopen their office locations, and corporate leaders discern the next phase of how work is structured. The data comes from aggregate telemetry from applications and application programming interfaces (APIs) protected by the Contrast Application Security Platform. It includes data on vulnerabilities, attacks, and open-source libraries between June 2020 and May 2021. Such comprehensive reporting from across the software development life cycle (SDLC) is only available from Contrast.

78% OF ACTIVE CODE IS CUSTOM

The fact that 70% or more of the code in the typical application comes from open-source libraries and frameworks has been widely noted in recent years. What is not usually revealed in these reports is that the vast majority of this code is not used by the application. Our data shows that 80% of code is open source, but only 6% of code belongs to an active library or class. The other 74% is from inactive libraries or inactive classes within active libraries. When this is taken into account, 78% of active code in the typical application is custom code and only 22% comes from open sources.

APPLICATION MAKEUP

20%

Custom Code

6%

Active Library Classes

74%

Inactive Libraries and Inactive Classes Within Active Libraries

FALSE-POSITIVE RATE

54%

for Critical CVEs

49%

for Major CVEs

Contrast SECURITY

39% OF CUSTOM CODE VULNERABILITIES ARE SERIOUS

Overall, a greater share of applications had custom code vulnerabilities than in the prior 12-month period. Most concerning is the 21% increase in the percentage of applications with at least one serious vulnerability—34% this year compared with 26% last year. Overall, 13% of applications have one or two serious vulnerabilities, while 3% have more than 100. And 39% of all vulnerabilities identified are serious, defined as High or Critical—a 39% increase over last year's 28% figure.

The top two vulnerability types in terms of percentage of applications affected are broken access control and cross-site scripting (XSS). Unfortunately, both of these vulnerability types were found in a larger share of applications this year than last year. As in the past, more Java applications have serious vulnerabilities than .NET ones—44% versus 23%.

34%

of Applications Have a Serious Vulnerability (21% increase over last year)

39%

of Vulnerabilities Are High or Critical

18%

of Applications Impacted by Broken Access Control and Cross-site Scripting Vulnerabilities

13%

of Applications Have 1 to 2 Serious Vulnerabilities; 3% Have 100+

VULNERABILITY ESCAPE RATE (VER) FALLS FROM 18 TO 1 IN 12 MONTHS

The good news is that Contrast enables developers to improve the security of their coding over time. With its immediate feedback when a vulnerability is created, the platform provides actionable instructions on how to fix the problem—and avoid making the same mistake again in the future. The result is akin to "just-in-time" security training—something that learning managers have struggled to provide for developers for years.

This phenomenon shows up in our data in what we are calling the vulnerability escape rate (VER). In the first two months of an application's tenure on the Contrast Application Security Platform, an average 12 vulnerabilities—six of them serious—occur in the software. But these numbers shrink steadily, reaching 6 total vulnerabilities and 3 serious in the ninth month. By the end of the first year, the average application sees no new serious vulnerabilities and just one non-serious one each month—a 92% decline.As in the past, more Java applications have serious vulnerabilities than .NET ones—44% versus 23%.

12 TO 1

The average vulnerability escape rate fell from 12 per month to 1 per month in 12 months—92% reduction

54% OF CRITICAL CVES ARE FALSE POSITIVES

Our data on application makeup, discussed above, upends a commonly held view of application security risk, as vulnerabilities that exist within the 74% of code that is inactive pose no risk to an organization. Traditional software composition analysis (SCA) tools do not differentiate between active and inactive code. As a result, every Common Vulnerability and Exposure (CVE) they identify that occurs in inactive code is actually a false positive. In our dataset, a majority of vulnerabilities—including 54% of CVEs rated as Critical and 49% rated as Major—would be false positives with traditional tools.

Nevertheless, good library hygiene such as keeping versions relatively up to date is important for reducing application security risk. Unfortunately, 45% of libraries found in applications are more than two years old.

FALSE-POSITIVE RATE

54%

for Critical CVEs

49%

for Major CVEs

MEDIAN TIME TO REMEDIATION NEARLY 29X BETTER THAN TRADITIONAL APPSEC SOLUTIONS

The Contrast customers in the dataset saw much better remediation timelines than organizations using legacy application security tools. The median time to remediation—the time it takes to resolve half of all closed vulnerabilities—was just 3 days in our dataset. This was 29 times faster than the 86 days reported by a traditional static application security testing (SAST) vendor. Nearly three-quarters (74%) of serious vulnerabilities were remediated within 90 days, and 90% of them within a year.

50% OF FIXED VULNERABILITIES CLOSED

3 DAYS

Contrast Customers

86 DAYS

Traditional SAST

SECURITY DEBT PER APPLICATION 19% LOWER

The result of this astounding remediation timeline is that organizations are reducing their per-application vulnerability backlog to 21 vulnerabilities, down from 26 in the previous 12 months. This 19% decline translates into lower application risk.

SECURITY DEBT

21 **Per Application: 21 Vulnerabilities, Down From 26**

99% OF ATTACKS WERE PROBES

The typical organization was pummeled with 25,343 application attacks every month over the past year. But one of the biggest trends in the attack data for the past 12 months is that the percentage of attacks that were viable was cut in half—from 2% last year to 1% this year. A viable attack is one that hits an existing vulnerability in an application, and therefore has a chance of being successful. Eleven of the top 12 attack types impacted a larger share of applications this year than last year, with big increases in broken access control, SQL injection, XSS, command injection, and expression language (EL) injection.

1% **of Attacks Were Viable, Down From 2%**

Vulnerability types with attacks that rose by 9+ percentage points:

BROKEN ACCESS CONTROL, SQL INJECTION, XSS, COMMAND INJECTION, EL INJECTION

RISKSCORE INDEX DOWN BY 1.25 POINTS ON AVERAGE

The report also updated the Contrast RiskScore Index, a numerical score that helps organizations rank and visualize the risk posed by different vulnerability types over time. The five highest RiskScores were quite consistent over the entire 12 months, but the average RiskScore declined by more than 1.25 points over the course of a year. This is because our data comes from Contrast Security customers, whose risk level declines as they build tenure with the solution (viz., the number of vulnerabilities being introduced declines and overall security increases). Vulnerability types that saw especially big declines include four types of injection—SQL, hibernate, NoSQL, and expression language (EL).

TOP 5 RISKSCORES (ANNUAL AVERAGES)

BROKEN ACCESS CONTROL, 9.45

CROSS-SITE SCRIPTING, 8.13

INSECURE CONFIGURATION, 7.36

SQL INJECTION, 7.07

SENSITIVE DATA EXPOSURE, 6.80

TAKEAWAYS

Findings in this report show that it is increasingly important for organizations to address software vulnerabilities in a timely manner, reduce their security debt, and prioritize their application security efforts according to actual risk. This is impossible to achieve without detailed observability on vulnerabilities, attacks, and open-source usage and security. Without it, organizations will not only waste scarce staff time on vulnerabilities that pose no risk but they may also defer action on truly dangerous issues in their software.

Instrumentation enables effective prioritization with continuous scanning, immediate feedback, and full observability. Contrast customers have a fuller understanding of which code is used by the software, which routes are exercised within the application, and what vulnerabilities need prioritized attention. The result is faster vulnerability remediation, reduced security debt, and fewer new vulnerabilities in applications.

03 Introduction

03 | Introduction

Contrast Labs' second annual Application Security Observability Report seeks to provide comprehensive analysis of application security trends during the time frame of June 2020 to May 2021. The research uses a broad dataset, including vulnerabilities identified by Contrast Assess, attacks detected by Contrast Protect, and open-source library data gathered by Contrast OSS.

Contrast Security is the only organization that provides data and insights across these three areas of telemetry in a single report, covering the entire software development life cycle (SDLC) and every component of an application or application programming interface (API). The data is also broken down by programming language, application age, routes exercised, and more, providing the most granular view of the state of application security available in the marketplace.

The goal of this report is to help security, development, and operations teams refine and prioritize their application security efforts. Contrast Labs supplements this comprehensive annual report with bimonthly Application Security Intelligence Reports, industry-specific[3] and role-specific[4] research, an annual report focusing on open-source security,[5] and regular updates on specific issues in Contrast Security's AppSec Observer blog.

The COVID-19 pandemic was still in its early stages when the 2020 Application Security Observability Report was researched and published, and most would agree that the 12 months since then have been a year like no other. One thing that did not change was a significant emphasis on the application layer as an attack vector by cyber criminals and nation-state actors.[6] Massive application attacks like the hack against SolarWinds Orion software[7] highlighted the importance of application security—so much so that a recent White House executive order on cybersecurity placed significant emphasis on delivering secure software.[8]

04 Application Makeup: More Custom Code Than Assumed

04 | Application Makeup: More Custom Code Than Assumed

It is widely reported that a vast and growing majority of the lines of code in the typical application comes from third-party libraries and frameworks. A 2020 study found that 70% of all code in applications is from third-party sources.[9] Our data shows an even higher percentage of third-party code, with only 20% of code coming from custom sources and 80% from libraries (Figure 1).

However, this figure is completely useless in calculating the amount of risk posed by third-party code. This is because nearly three-quarters (74%) of code is inactive—that is, not invoked by the application at all. Nearly half of code is from inactive libraries—full libraries that are never used by the software but are included because they are attached to other libraries that are in use.

Even within active libraries, most of the code is never invoked: 25% of code in the typical application belongs to an inactive class within an active library. Active library classes, on the other hand, comprise just 6% of code. With custom code making up 20% of an application, narrowing the focus to active code changes the typical view of application security risk dramatically. Overall, third-party libraries make up just 22% of active code in the typical application, while custom code comprises 78%.

BY LANGUAGE

While our overall dataset shows just 6% of code consists of active library classes, the number is a bit higher for the two most common programming languages—Java and .NET. Active library code makes up 9% and 13% of those applications, respectively (Figure 2). This is because applications written in the Node language have especially low percentages of active third-party code. And a total of one-third of code (both custom and third party) is invoked in both Java and .NET. Of course, this means that two-thirds of code in these languages is still inactive.

Understanding which parts of an application are active is critical to properly prioritizing vulnerability remediation, as vulnerabilities in inactive code generally pose no risk to an organization. Yet, organizations spend multiple staff hours updating libraries that are never used by an application to remediate Common Vulnerabilities and Exposures (CVEs) that have been published. While keeping libraries up to date is a matter of good hygiene, it can actually increase risk if remediations that do not reduce risk are prioritized ahead of those that do.

FIGURE 1

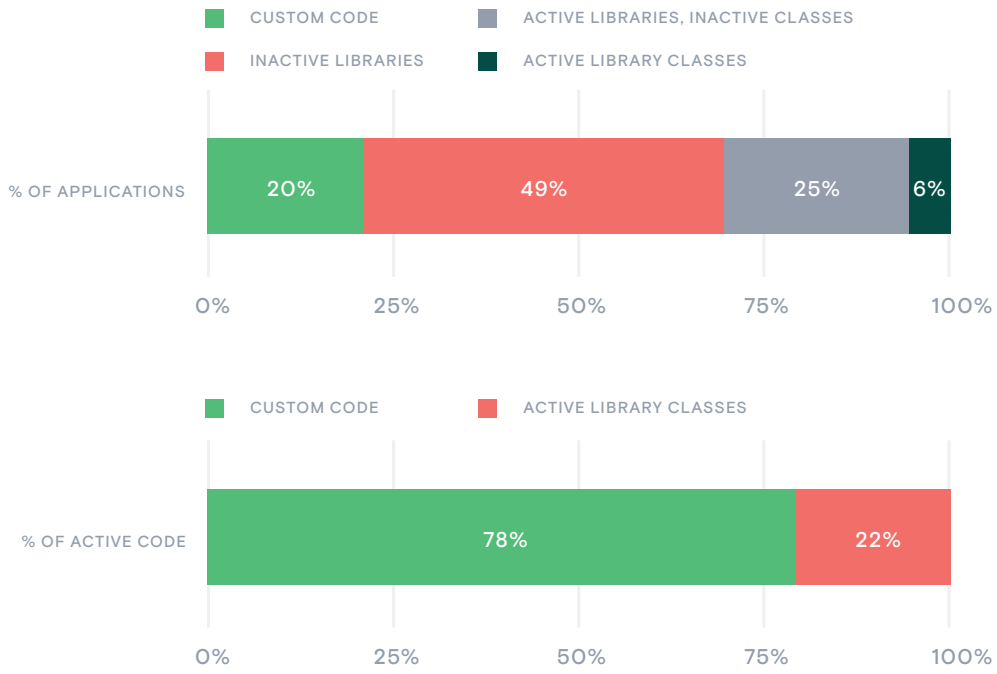
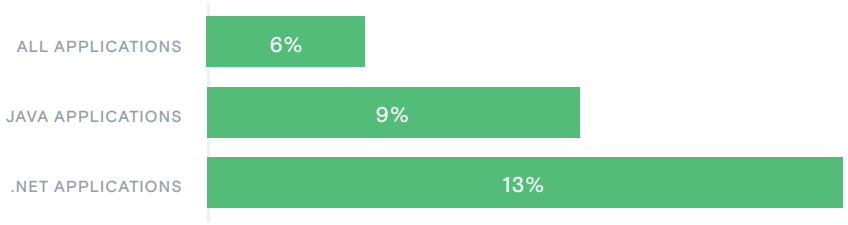
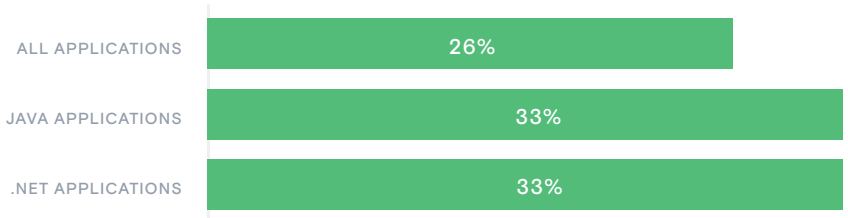


FIGURE 2

% of Applications Made Up of Active Library Code



% of Applications Invoked

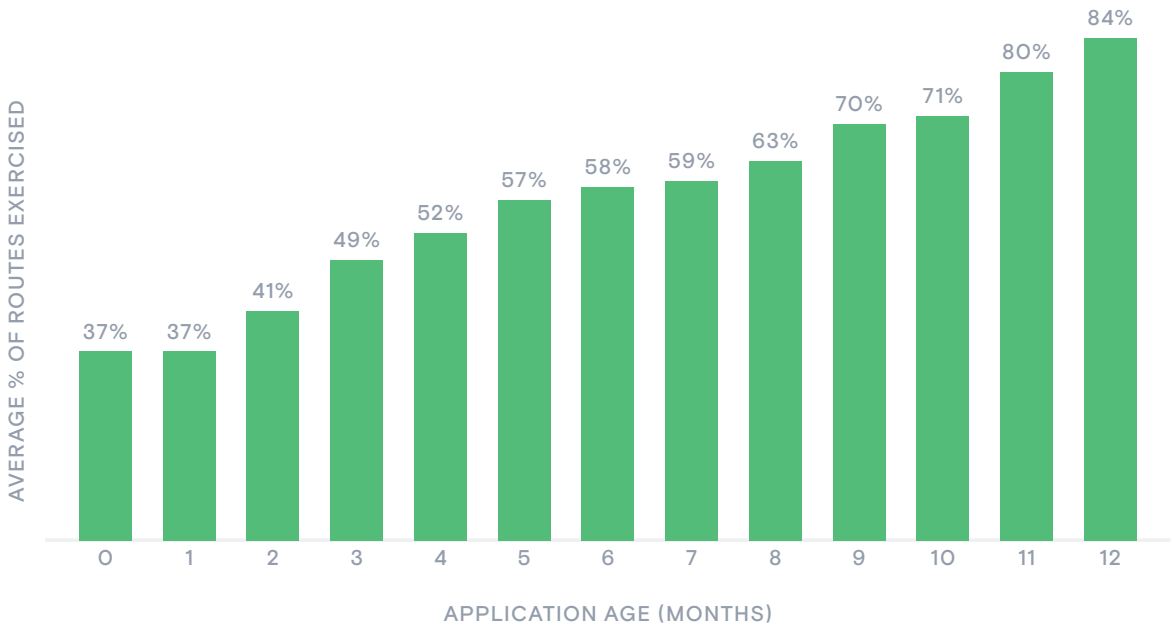


05 Routes: Vulnerabilities Decline Later in the SDLC

05 | Routes: Vulnerabilities Decline Later in the SDLC

The Route Intelligence functionality in Contrast Assess analyzes the different routes users can take when interacting with a piece of software. The functionality has been live for over a year, and it is interesting to observe the trends around which possible routes in an application are exercised. Across the entire dataset, the percentage of potential routes exercised by an application increases as its age increases (Figure 3). Only after the ninth month do more than half of applications have more than 80% route coverage (Figure 4). This makes sense for software under development, as some functionalities may not yet be in place in the earlier stages of the SDLC. By the time the typical application is 12 months old, more than 84% of potential routes are exercised.

FIGURE 3

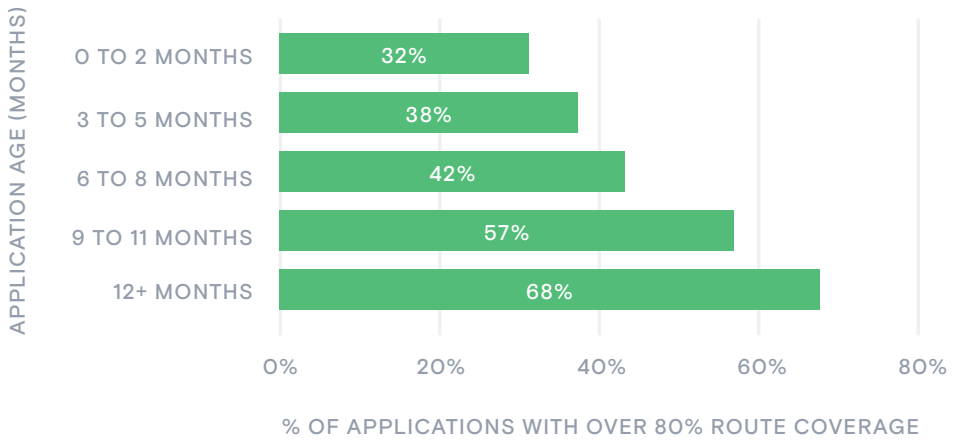


ROUTE INTELLIGENCE

An application route represents how a user interacts with an application. It consists of three distinct data points: the URL of the route, the HTTP verb associated with the request (e.g., GET or POST), and a unique signature based on that route's controller action. Contrast Security's Route Intelligence capability observes an application at runtime, rather than testing and retesting lines of code. Such monitoring reveals the different points of entry into the application, and detects vulnerabilities that become apparent as the code runs—whether it comes from custom or third-party sources.

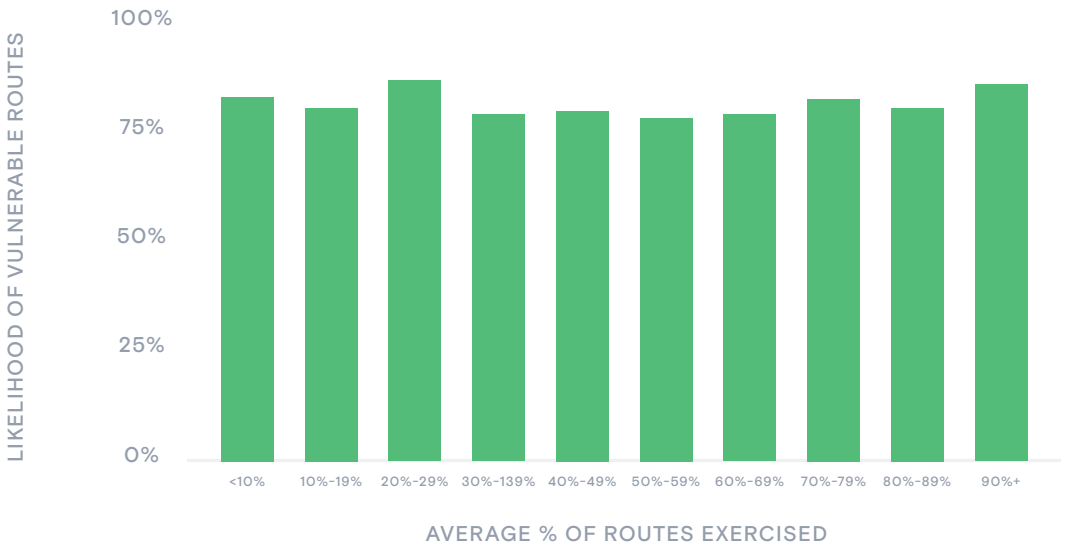
Beyond its ability to detect vulnerabilities that only manifest themselves in the running application, Route Intelligence also helps organizations with prioritization of application security efforts. Because it provides observability into which routes are invoked by the application, organizations can understand which vulnerabilities present risk, and which are on a route that a user would never reach.

FIGURE 4



Regardless of the percentage of routes exercised by the software, data shows that a similar percentage of routes contains at least one vulnerability. When the percentage of routes exercised is divided into 10-percentage-point groups, between 78% and 86% of routes are vulnerable (Figure 5). Major variances based on the percentage of routes exercised occur.

FIGURE 5



06 Custom Code Vulnerabilities: More Likely to be Serious

06 | Custom Code Vulnerabilities: More Likely To Be Serious

Overall, more applications had vulnerabilities and serious vulnerabilities in their custom code in the past 12 months than in the prior year (Figure 6). The percentage of applications with at least one vulnerability increased from 96% to 98% and the share with at least one serious vulnerability increased from 26% to 34%—a 30.7% increase. Among all vulnerabilities, nearly 4 in 10 (39%) are classified as serious—High or Critical (Figure 7). This is a 39% increase over the 28% that was observed last year.

FIGURE 6

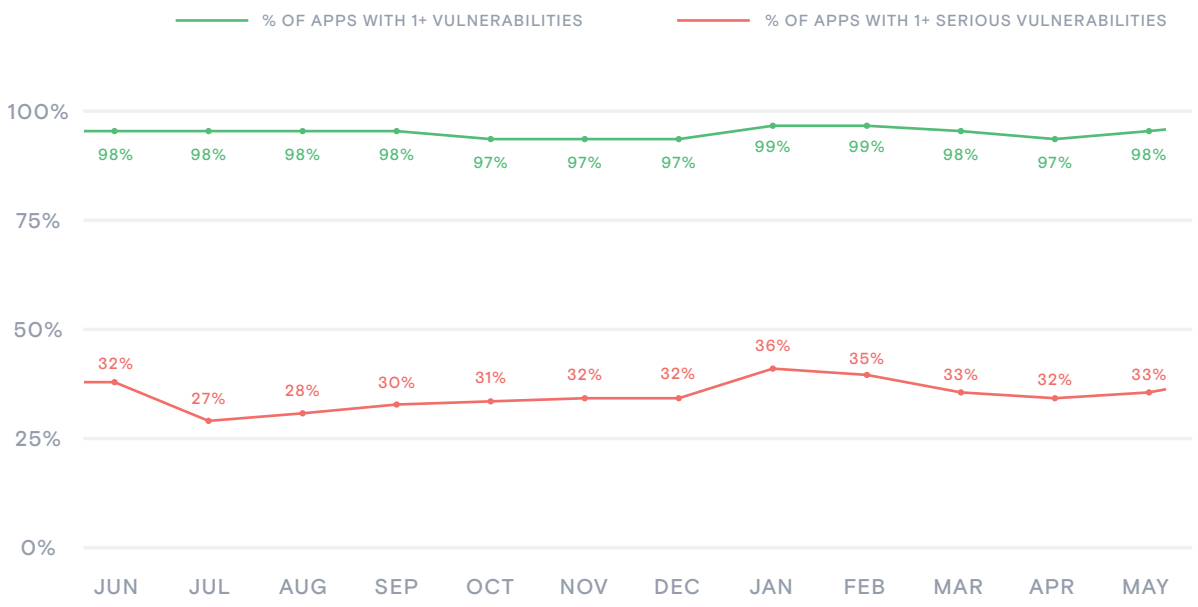
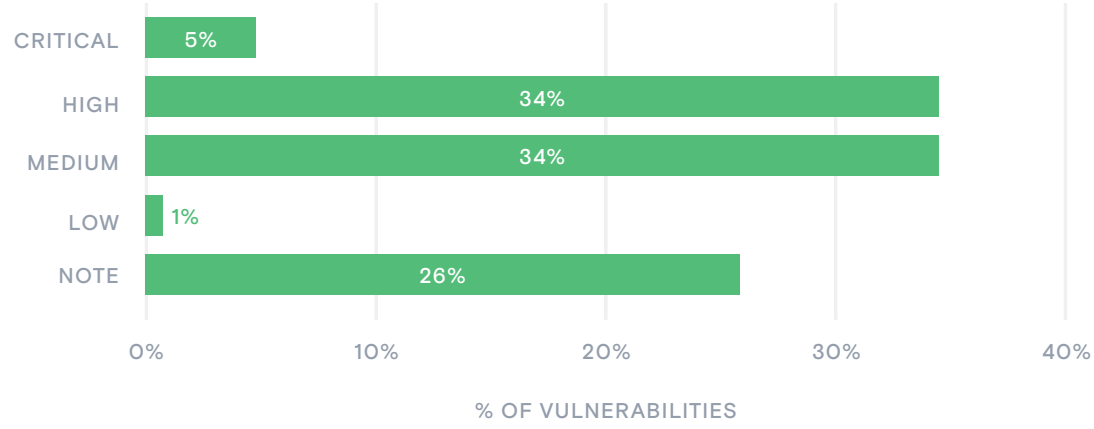
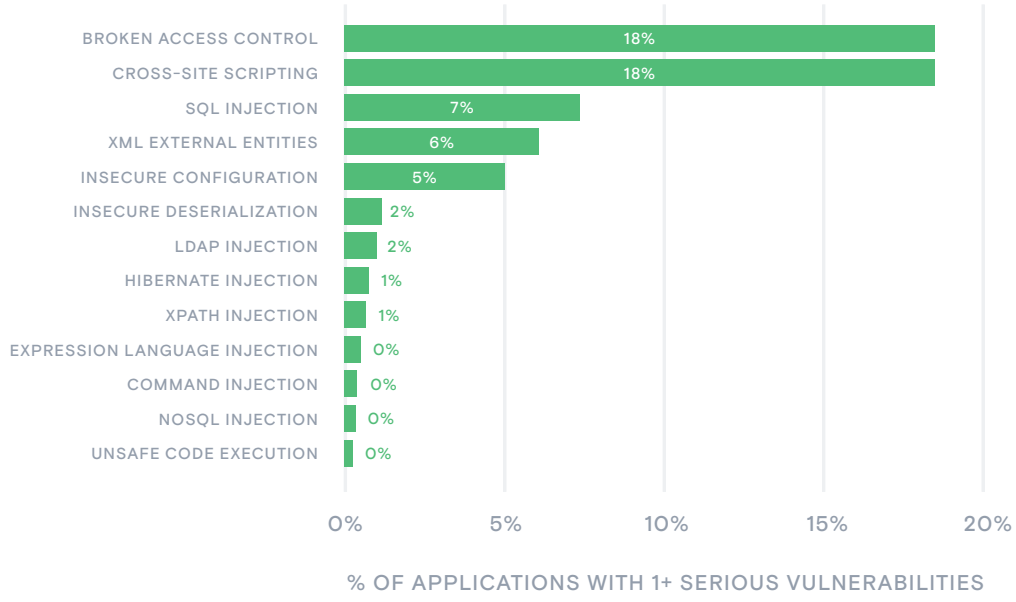


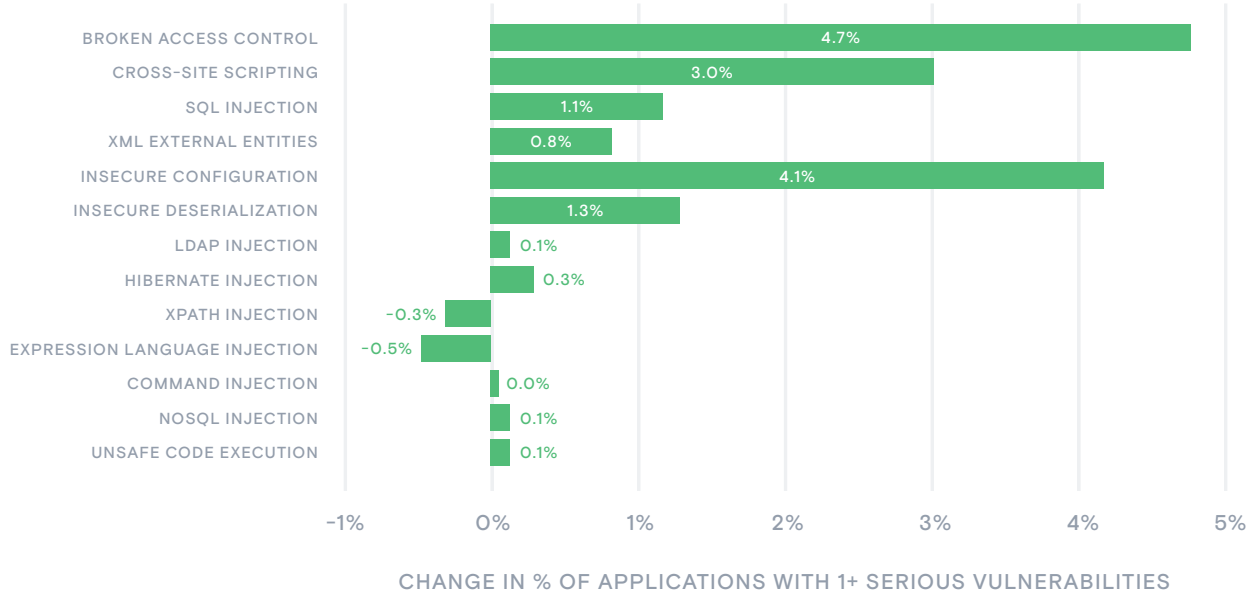
FIGURE 7



Looking at vulnerabilities by type, broken access control and XSS impact by far more applications than any other types (Figure 8). This in itself is not surprising, but the large increase in both is concerning. The percentage of applications with a broken access control vulnerability type increased from 13.3% to 18%—a 35.3% increase. XSS vulnerabilities were in 18% of applications, compared with 15% from the prior 12 months (June 2019–May 2020)—a 20% increase. A vulnerability type worth watching is insecure configuration, which increased by a delta of four percentage points, quintupling from impacting 1% of applications in the prior year to 5% this year.

FIGURE 8

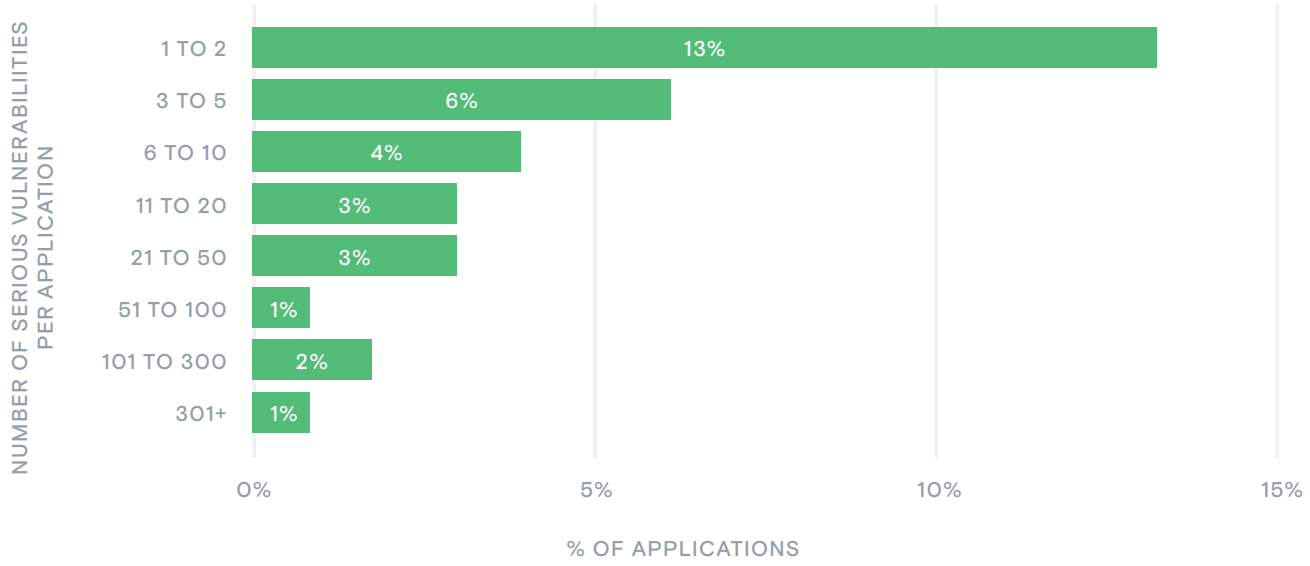




VULNERABILITIES PER APPLICATION

A subset of applications in the dataset has a large number of vulnerabilities—including 1% of applications that have more than 300 (Figure 9). This skews the averages upward. One in 10 applications has more than 10 vulnerabilities, and only 13% have as few as one or two.

FIGURE 9



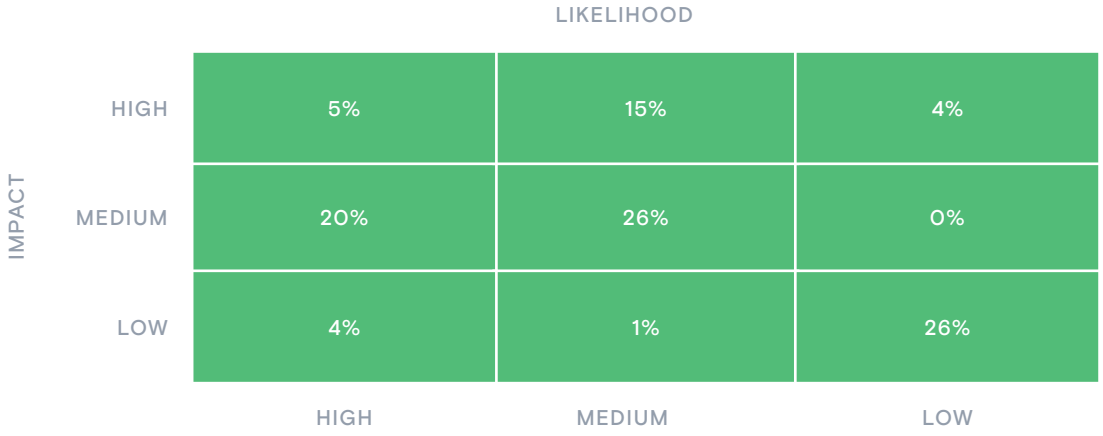
BY LIKELIHOOD AND IMPACT

It is instructive to look at vulnerabilities by both how likely they are to occur and the impact when they do occur. Contrast Labs rates each vulnerability type for the perceived technical and business impact of a successful exploit. SQL injection, for example, is rated as a high-impact vulnerability type because it leads to a complete takeover of a database, complete data exfiltration, and a complete degradation of confidentiality, integrity, and availability of the database.

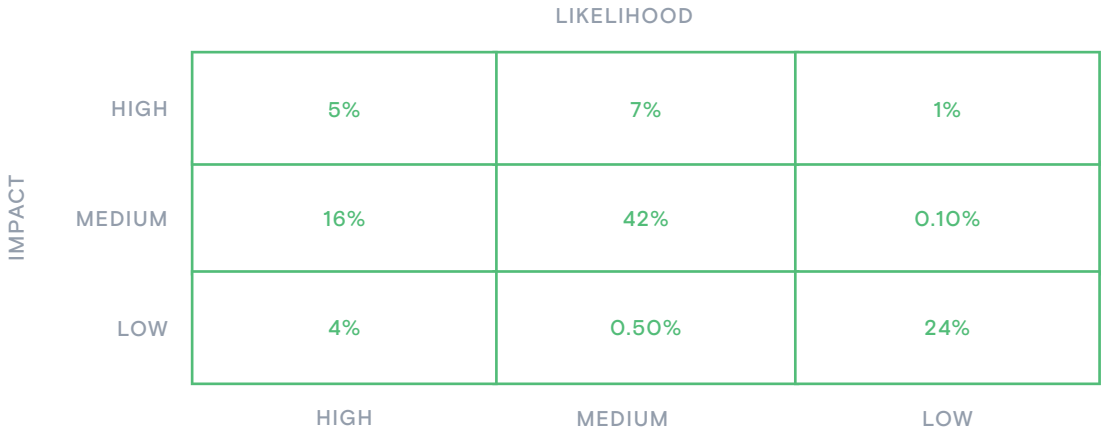
In the most recent 12-month period, 5% of vulnerabilities are both high likelihood and high impact—and any number above zero is too high for this scenario (Figure 10). But another 20% of vulnerabilities are high impact with medium likelihood, while another 15% are medium likelihood and high impact. These three figures add up to 40% this year, compared with 28% in the prior 12-month period. Organizations, as a result, need to prioritize remediation on these vulnerabilities over those with lower likelihood and impact.

FIGURE 10

This Year



LastYear



BY LANGUAGE

The overall increase in serious vulnerabilities is driven by an increase in .NET serious vulnerabilities. While the percentage of Java applications with at least one serious vulnerability increased from 42% to 44%, .NET applications saw an increase in this figure from 16% to 23%—a 44% increase (Figure 11).

FIGURE 11

JAVA

	LAST YEAR	THIS YEAR	DELTA
% OF APPLICATIONS WITH AT LEAST 1 VULNERABILITY	97%	98%	1%
% OF APPLICATIONS WITH AT LEAST 1 SERIOUS VULNERABILITY	42%	44%	2%

.NET

	LAST YEAR	THIS YEAR	DELTA
% OF APPLICATIONS WITH AT LEAST 1 VULNERABILITY	97%	99%	2%
% OF APPLICATIONS WITH AT LEAST 1 SERIOUS VULNERABILITY	16%	23%	7%

Despite Java remaining relatively the same, one area was particularly troubling: The percentage of Java applications with insecure configuration vulnerabilities tripled (Figure 12). In .NET applications, XPath injection vulnerabilities impacted 3% of applications this year, compared with just 1% last year (Figure 13)—a 300% increase.

FIGURE 12

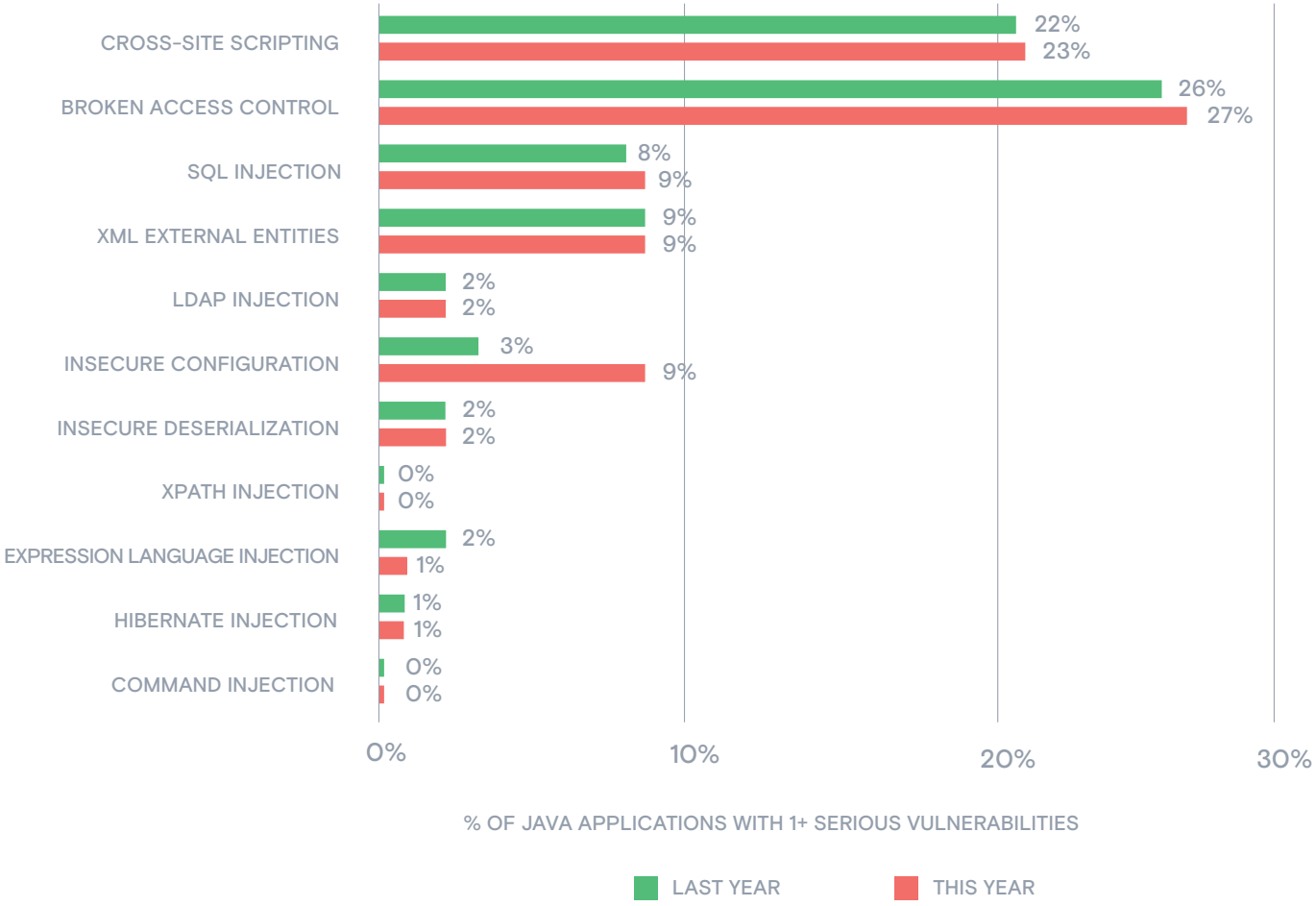
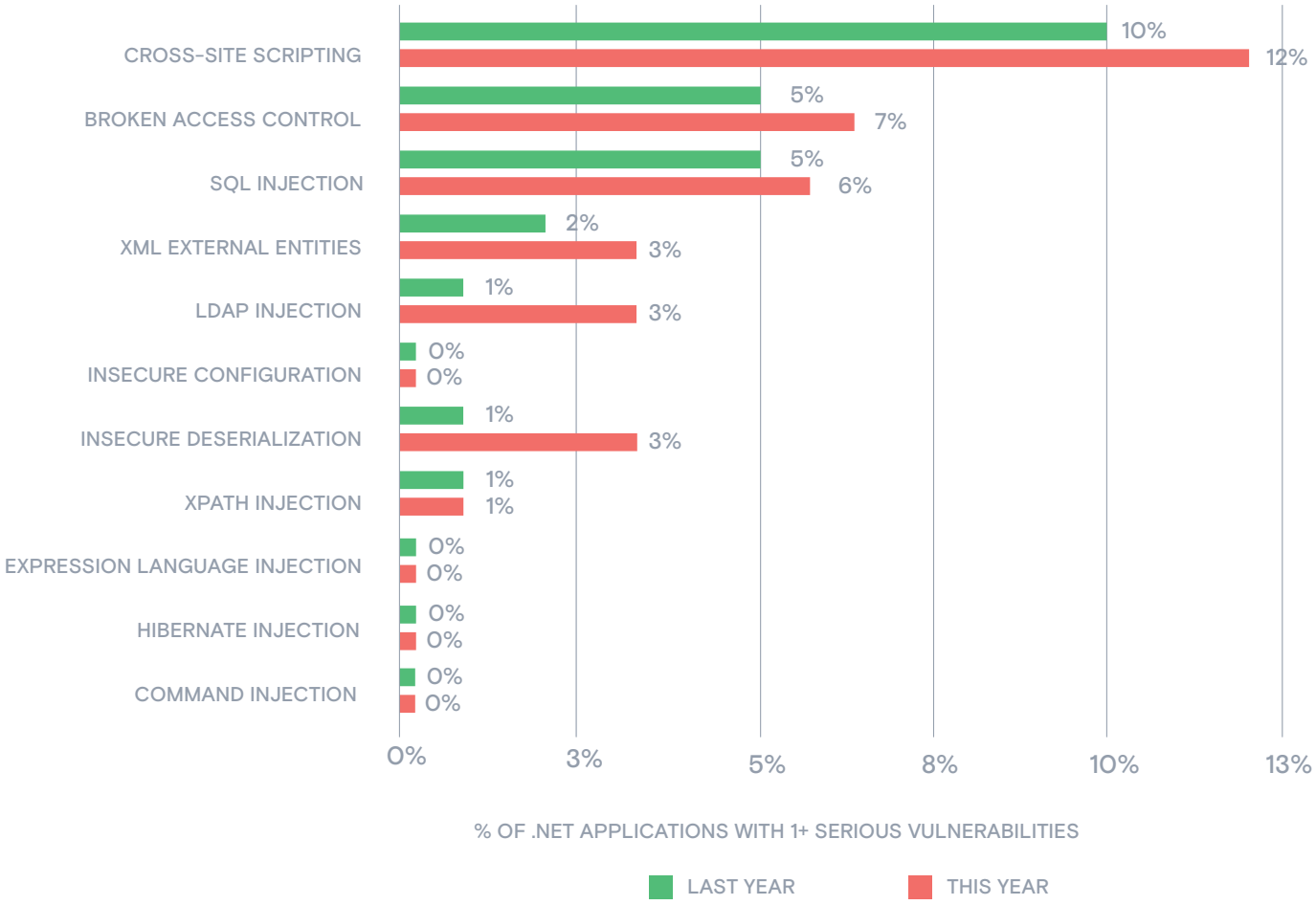


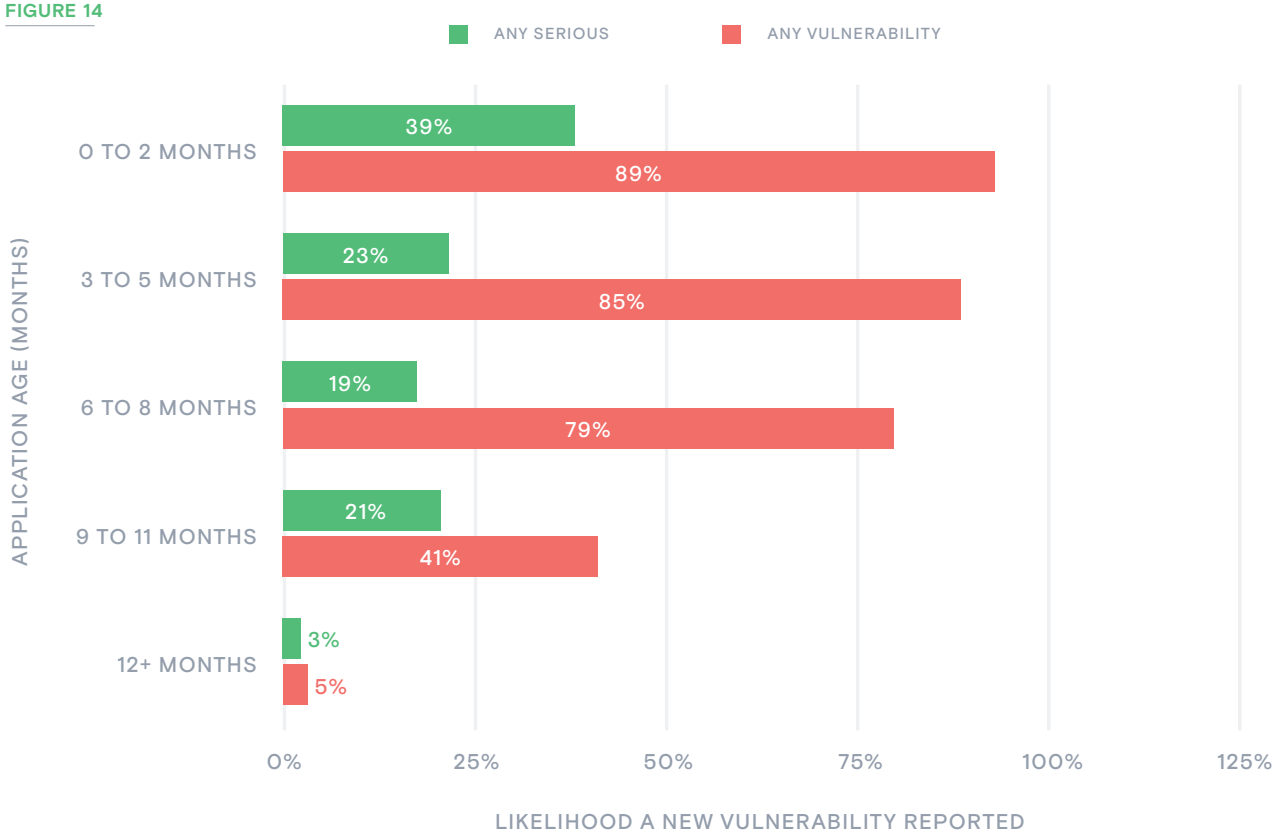
FIGURE 13



BY APPLICATION AGE

Looking at vulnerabilities by application age, it becomes clear that the Contrast customers in our dataset are reducing the number of new vulnerabilities that occur as they work their way through the SDLC. While a new application (0–2 months) has a 39% chance of seeing a new serious vulnerability, an application that is more than a year old only has a serious vulnerability 3% of the time (Figure 14).

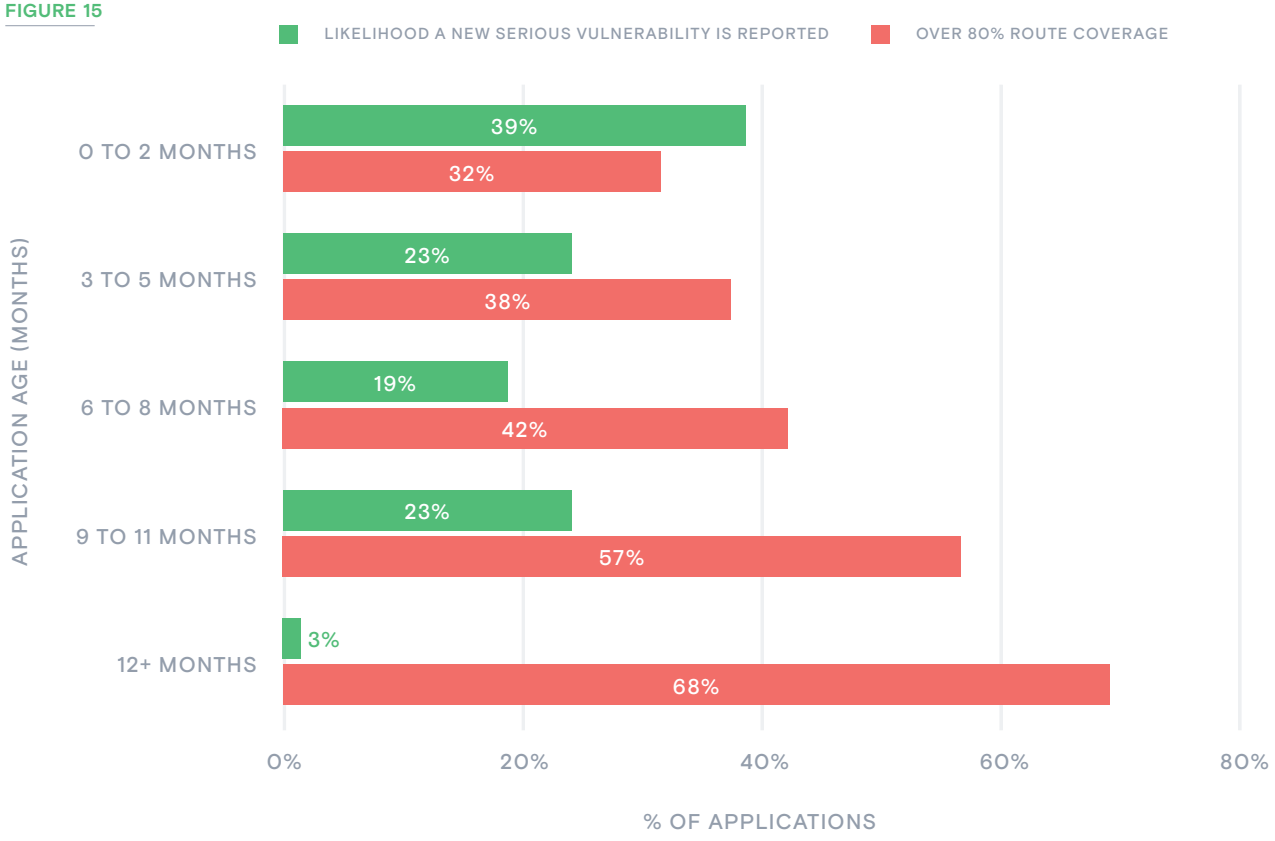
FIGURE 14



FOR APPLICATIONS WITH 80%+ ROUTE COVERAGE

It is interesting to compare these vulnerability figures with the route coverage data we discussed earlier in the report. As an application gets older, more potential routes are invoked by the software, potentially increasing the attack surface. But the likelihood of a new vulnerability declines (Figure 15).

FIGURE 15



VULNERABILITY ESCAPE RATE (VER)

One interesting insight revealed by the vulnerability data is that the Contrast customers in our dataset see dramatic improvements in vulnerability prevalence quite quickly after onboarding an application into the Contrast Application Security Platform. In the first two months of an application's tenure on the platform, an average of 12 vulnerabilities—six of them serious—occur in the software (Figure 16). These numbers shrink steadily, reaching 3 and 6 after nine months. By the end of the first year, the average application has a vulnerability escape rate (VER) of no new serious vulnerabilities and just one non-serious one.

Why does this happen? Users of the Contrast platform receive immediate feedback when a vulnerability occurs, along with contextual information on how to fix it. This provides "just-in-time" training to developers, helping them to avoid that same mistake in the future. After a few months, most developers will have gone through this process for most of the common vulnerability types, resulting in fewer new vulnerabilities. Shorter feedback loops and high levels of accuracy help developers write better code over time. Ultimately, this contributes to greater velocity for the development process and fewer security-related delays.

FIGURE 16


ANY SERIOUS
ANY VULNERABILITY
APPLICATION AGE (MONTHS)
0 TO 2 MONTHS
3 TO 5 MONTHS
6 TO 8 MONTHS
9 TO 11 MONTHS
12+ MONTHS
6
12
4
8
4
7
3
6
0
1
0
5
10
15%
AVERAGE NEW VULNERABILITIES REPORTED

07 Open-Source Libraries And Frameworks: Organizations Struggle To Keep Up

07 | Open-Source Libraries and Frameworks: Organizations Struggle to Keep Up

As noted earlier, our data shows that 80% of code in the typical application comes from open-source libraries and frameworks, but such third-party sources only produce 22% of active code. Beyond the significant time savings achieved by using already-written code rather than "reinventing the wheel" with custom code, open-source libraries also have the advantage of an army of security researchers who identify CVEs that can be remediated with version updates to the libraries. In actuality, this may be seen as both a blessing and a curse, as the number of CVEs identified in recent years has exploded, potentially overwhelming security teams that are attempting to keep up.[10]

Contrast Labs understands the importance of open-source code to modern development teams, and devotes significant research each year to helping organizations keep their libraries and frameworks secure, including publishing an annual State of Open-source Security Report.[11]

ACTIVE LIBRARIES AND CLASSES

One pain point for many security and development teams centers on a lack of visibility into which open-source libraries and classes are active versus inactive. Over the past 12 months, the average application contained just over 125 libraries, 48 of which were active (Figure 17)—or only 38.4%. Among active libraries, only 46 out of 163 library classes are active (Figure 18)—or 28.2%. Organizations without a clear understanding of which code is actively used by the application will waste resources remediating CVEs that pose no risk because they are in an inactive library or class.

FIGURE 17

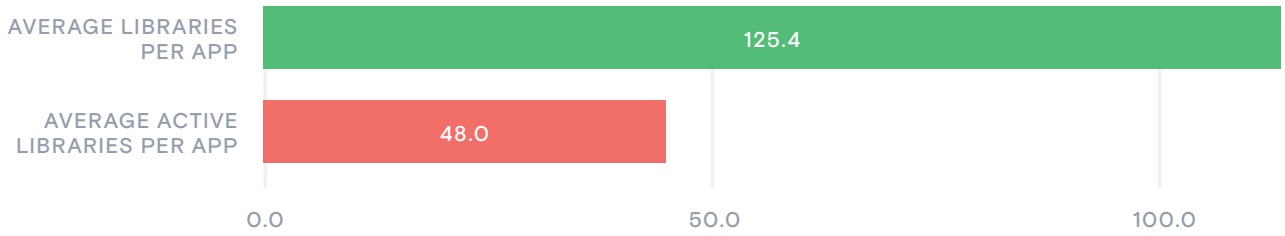
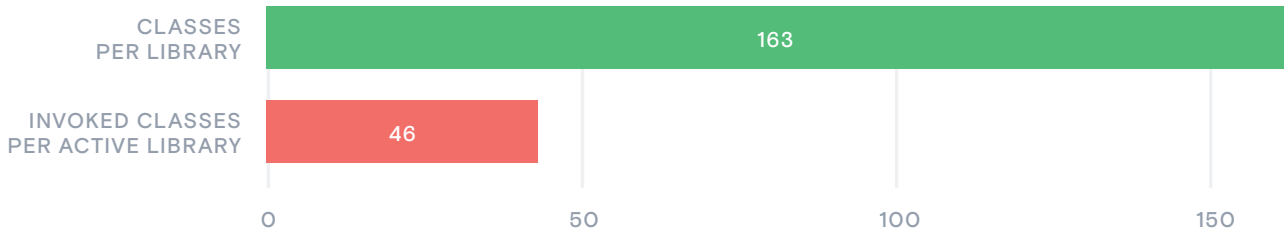


FIGURE 18



Contrast SECURITY

It should be noted that these library count statistics are means, and the medians are somewhat lower due to a subset of applications that contain a large number of libraries. More than half of libraries have fewer than 50 libraries, and nearly half have fewer than 25 active libraries (Figures 19 and 20).

FIGURE 19

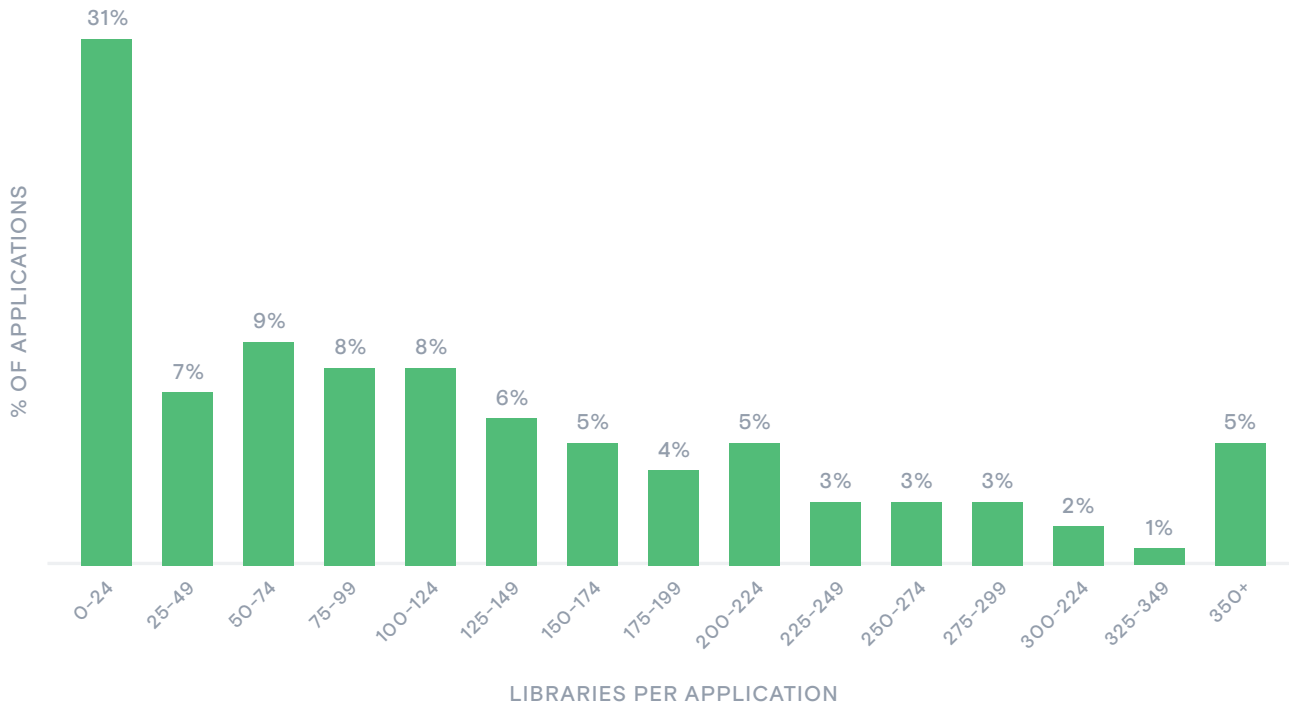
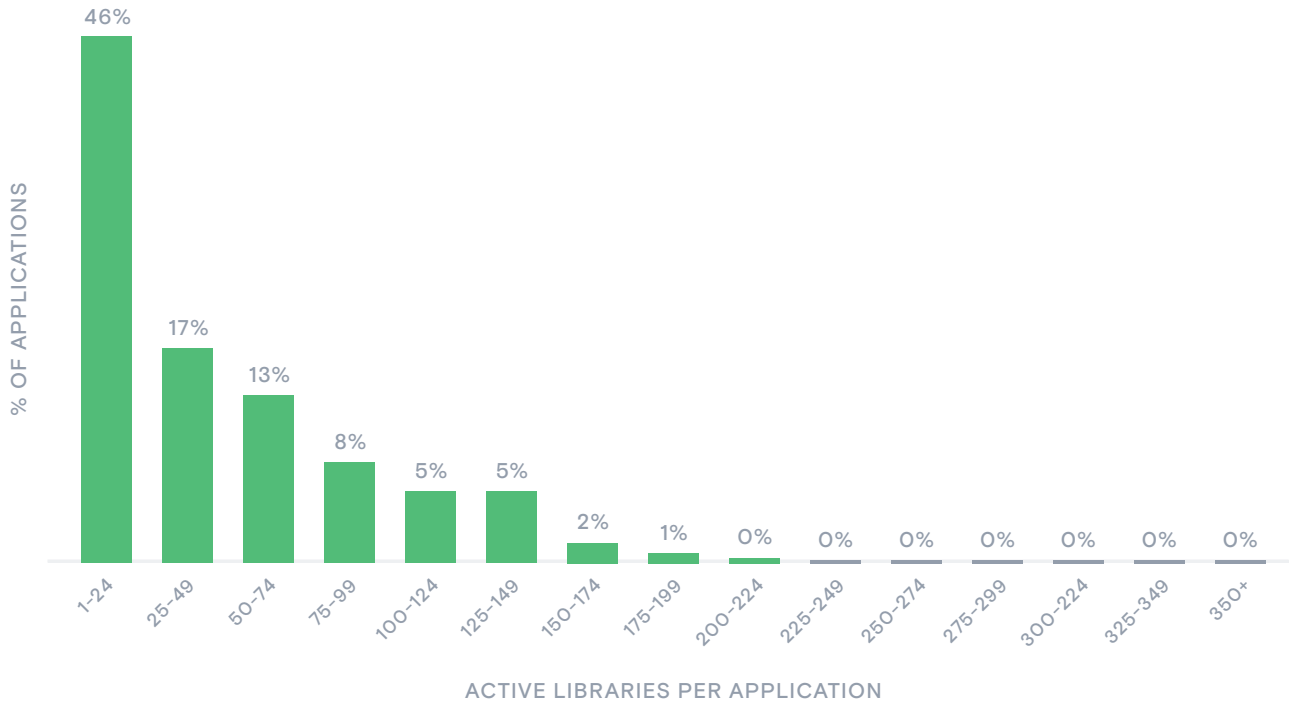


FIGURE 20



OPEN-SOURCE VULNERABILITIES

When it comes to unfixed CVEs in applications, it is instructive to look at the entire universe of libraries compared with active libraries in applications. For all libraries, 3% of libraries have CVEs that are serious (Critical or Major; Figure 21). The same percentage of active libraries have serious CVEs, but there are fewer Critical ones and more Major ones (Figure 22).

FIGURE 21

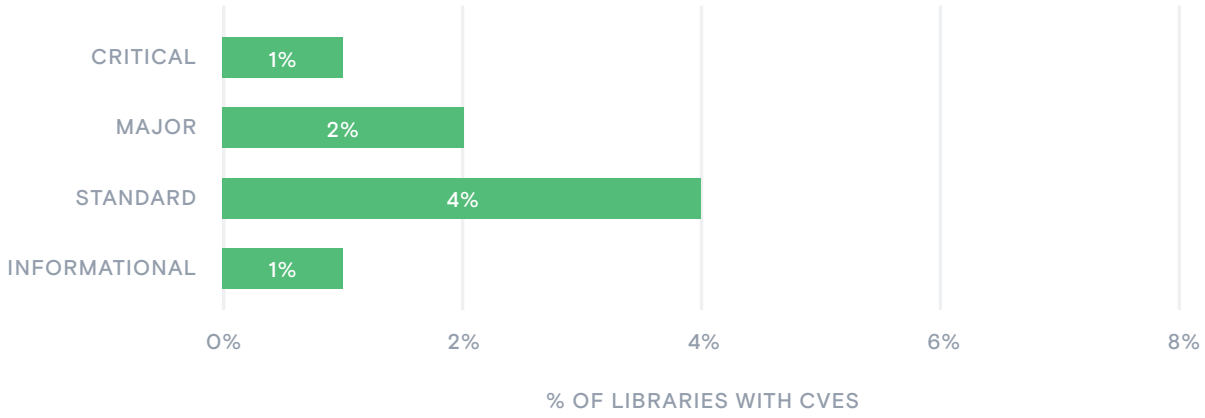
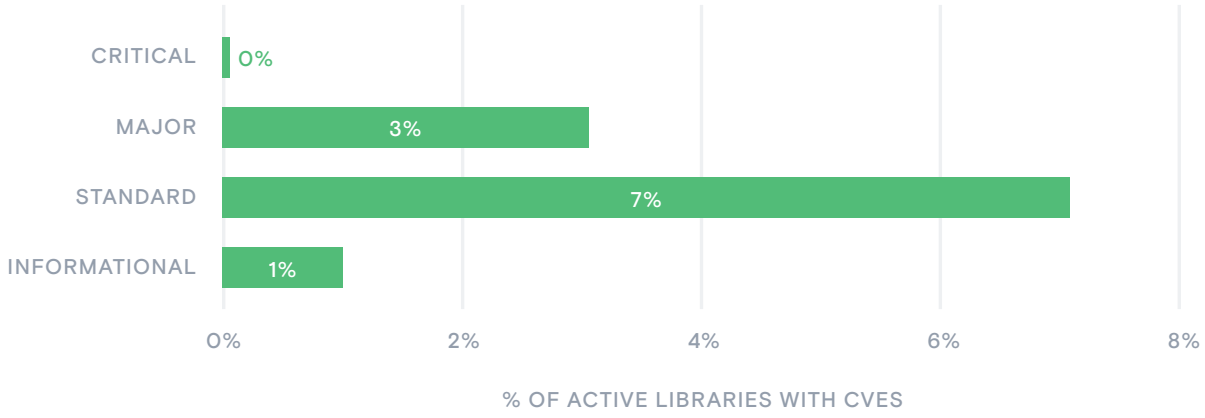


FIGURE 22



MOST CVES ARE FALSE POSITIVES

Contrast OSS users can easily determine which CVEs are present in inactive libraries and library classes, and therefore pose no risk to the organization. They can then deprioritize these CVEs for remediation or library update. Traditional software composition analysis (SCA) tools that simply return a list of CVEs produce a false positive every time they list a CVE in an unused part of the software.

Our data shows that a majority of CVEs found in the applications are in inactive libraries or library classes, and would therefore generate a false positive with traditional tools (Figure 23). This includes 54% of CVEs rated as Critical and 49% of those rated as Major, which would be high remediation priorities for those using traditional tools. While keeping libraries up to date is seen as a best practice, there is a risk that teams will spend time updating libraries or doing other remediation for CVEs that pose no risk while potentially delaying action on more risky vulnerabilities. Given that security teams receive an overwhelming number of security alerts every day, it is important to have enough observability to effectively prioritize these efforts.

FIGURE 23

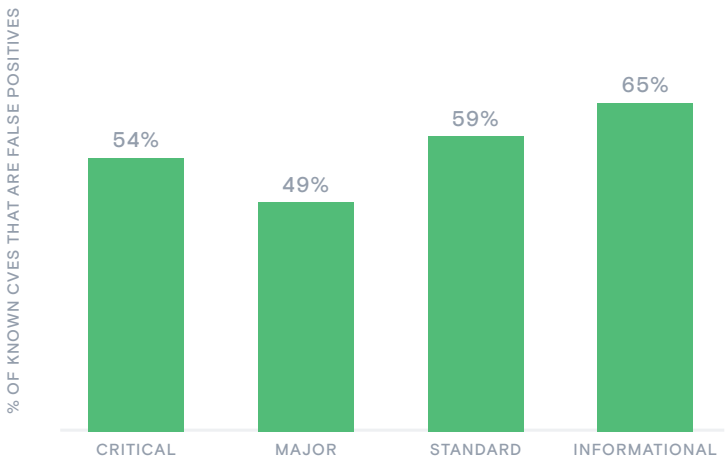
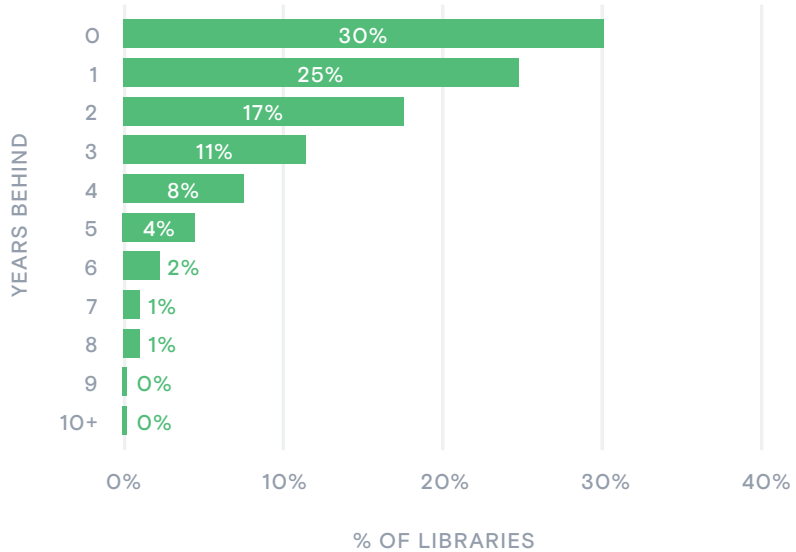


FIGURE 24



LIBRARY AGE: OLDER LIBRARIES CAN COMPOUND RISK

The main method for remediating CVEs as they are published is to update the version of the library or framework to a later version that includes a fix for the specific vulnerability. Thus, it is important to understand library age. Since different libraries have different update frequencies and version numbering conventions, the best way to get an "apples-to-apples" comparison is to measure the time elapsed since a library version was released.

While 30% of libraries found in applications use versions less than a year old, 45% are two or more years old (Figure 24). While the risk posed by older libraries depends on how active they are, the fact that so many old libraries exist in applications is a concern.

08 Security Debt: Organizations are Making Progress

08 | Security Debt: Organizations are Making Progress

Unresolved vulnerabilities tend to accumulate, and many organizations amass hundreds or thousands of them over time. This security debt is a burden on development and security teams, which makes it harder to continue producing functional and secure software. In addition, security debt increases risk. Recent application attacks on Microsoft Exchange Server and Kaseya VSA exploited vulnerabilities that had been discovered but not made public. In each case, development of the patch was in progress when the attacks occurred.[12]

In our dataset, security debt per application declined from 26 to 21, a 19.2% decrease (Figure 25). For security debt per organization, it slightly increased (4.75%; Figure 25) over the past 12 months. This is troubling when combined with the fact that only 28% of organizations reduced their security debt in the same timeframe (Figure 26). However, when one factors into consideration the aforementioned jump in the number of applications protected by the Contrast platform, these numbers make complete sense.

FIGURE 25

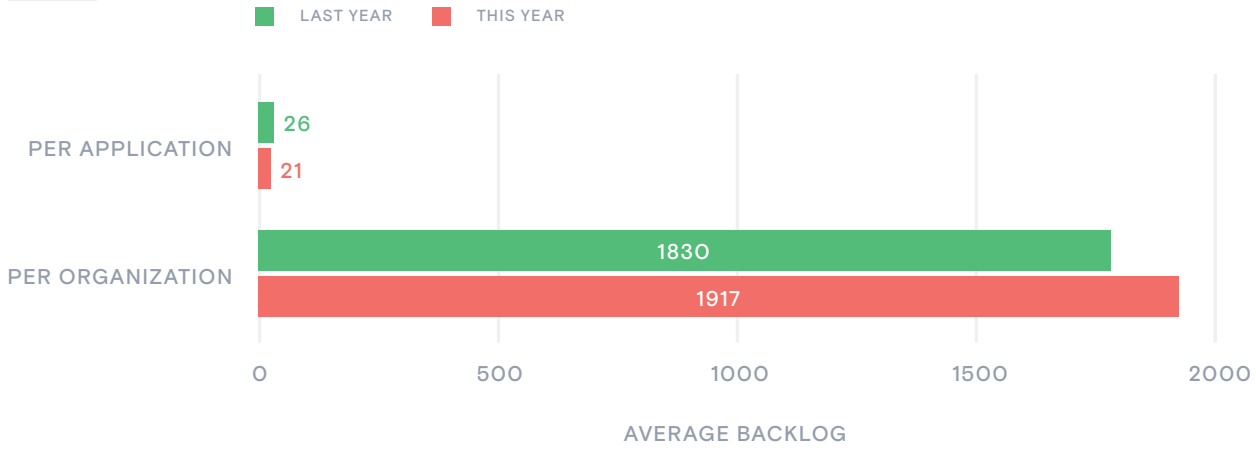
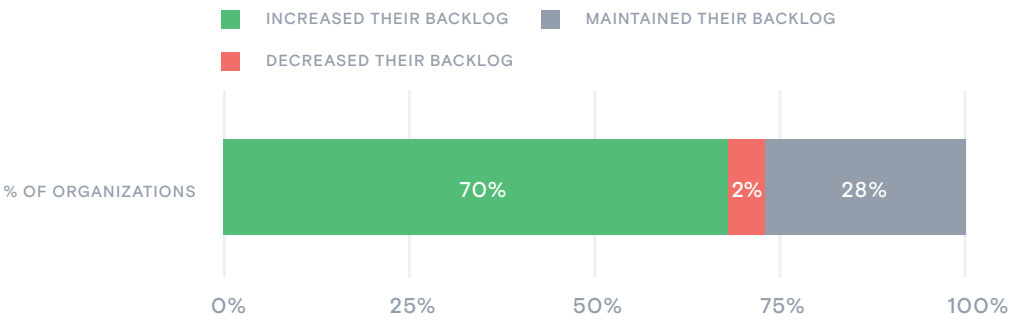


FIGURE 26



09

Vulnerability Remediation: Slower But Much Better Than Traditional Approaches

09 | Vulnerability Remediation: Slower But Much Better Than Traditional Approaches

Remediation timelines are an important consideration when measuring the effectiveness of application security efforts at an organization. When vulnerabilities are repaired months or years after they are created, it requires significantly more time, cost, and effort to dig through multiple layers of subsequently written code to fix the problem. The need to remediate vulnerabilities that have been put on the back burner can also delay the rollout of an application and can increase the likelihood that a vulnerability slips into production.

COMPARED WITH LAST YEAR

Our data for the past 12 months shows a mean time to remediation (MTTR) of 96 days, somewhat higher than last year. One factor in this increase is the addition of highly complex applications to the dataset since last year. In FY2021, for example, Contrast's active users increased by 150% and the number of applications being monitored increased by 190%. The sharp increase in applications in the dataset resulted in a 270% spike in the number of vulnerabilities recorded. As a result, newer customers and applications in the dataset are still working to catch up on remediation.

Overall, more than 6 in 10 (62%) vulnerabilities (Figure 27) and 74% of serious vulnerabilities (Figure 28) are remediated within 90 days. At the one-year mark, more than 90% of serious vulnerabilities and around 80% of all vulnerabilities have been addressed (Figure 29).

FIGURE 27

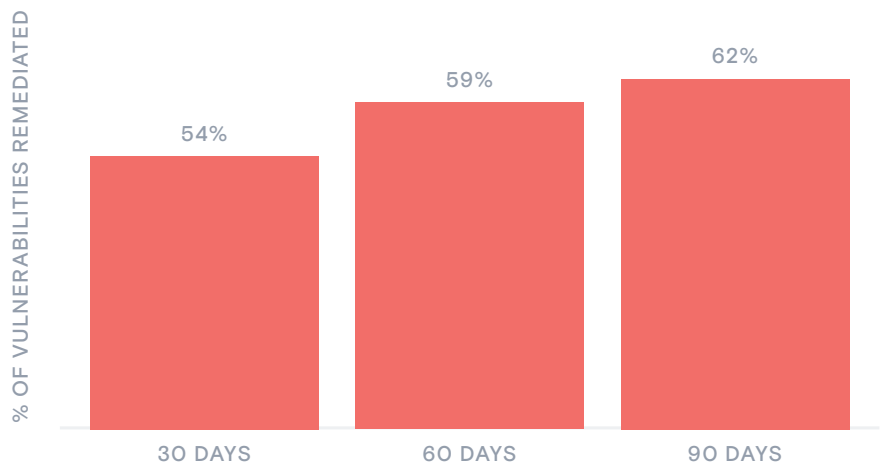


FIGURE 28

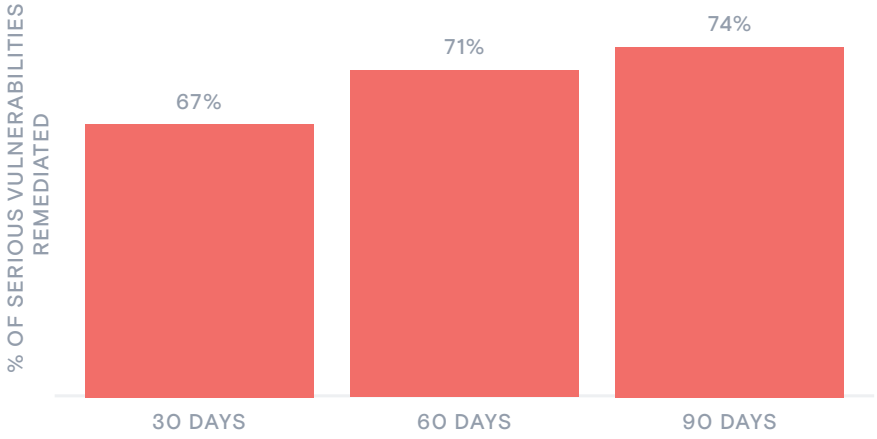
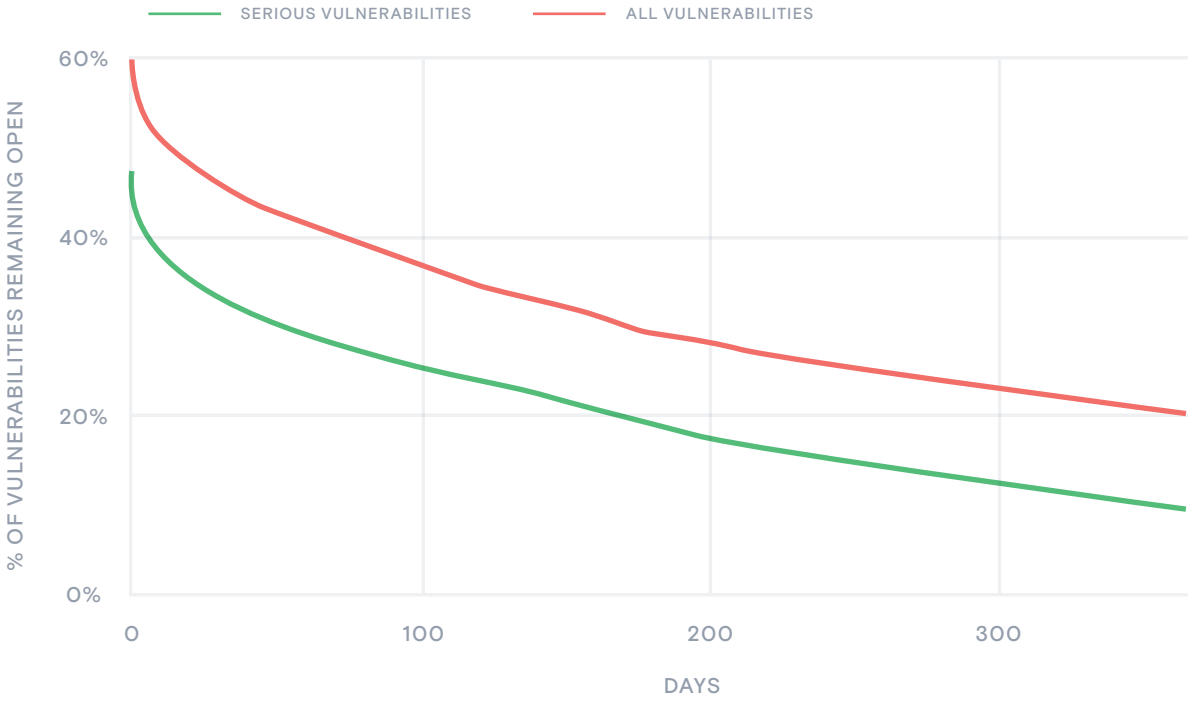


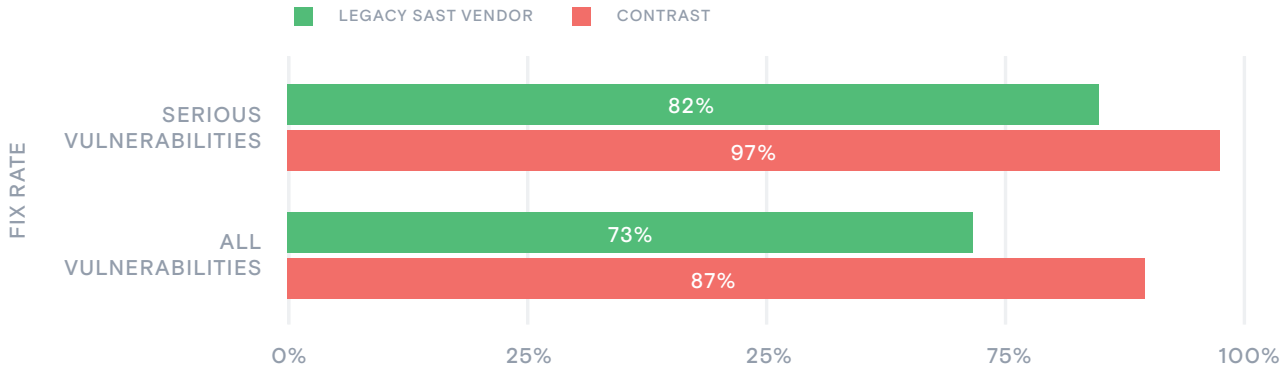
FIGURE 29



COMPARED WITH TRADITIONAL PRACTICES

The Contrast customers in our dataset fared significantly better than users of traditional application security solutions when it comes to the remediation timeline. We compared the outcomes in our dataset with similar analysis conducted by a traditional static application security testing (SAST) vendor.[13] Here, we found that the fix rate for serious vulnerabilities bested that vendor's dataset by 15 percentage points—97% versus 82% (Figure 30).

FIGURE 30



The comparison is even more stark when looking at the median time to remediate—the amount of time it takes to take care of half of an application's vulnerabilities. For all vulnerabilities, that figure is 11 days for the Contrast customers in our dataset, while the SAST vendor reports 216 days (Figure 31)—19.6 times faster. Among vulnerabilities that are ultimately fixed, the numbers are 3 days versus 86 days—28.7 times faster. The time required to remediate 75% of vulnerabilities is 249 days for the Contrast dataset versus 540 days for SAST customers (Figure 32), slightly less than half the time (46%).

FIGURE 31

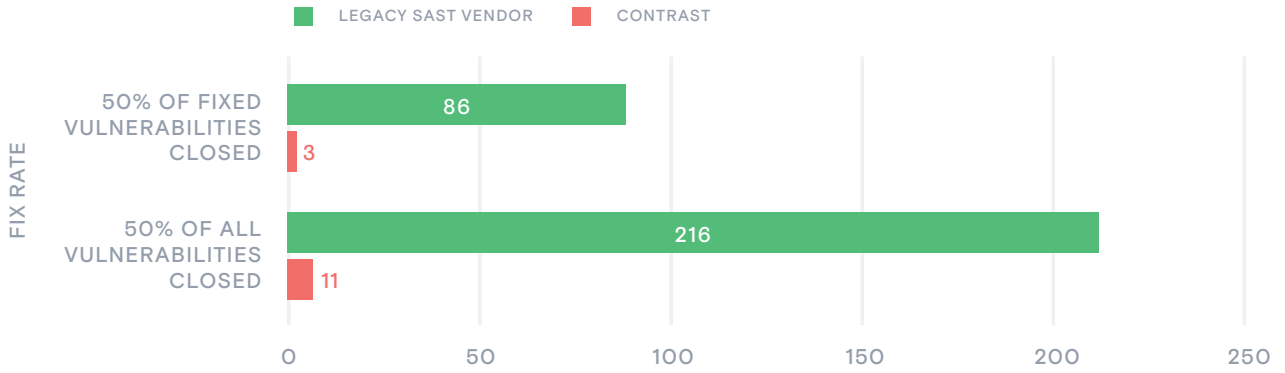
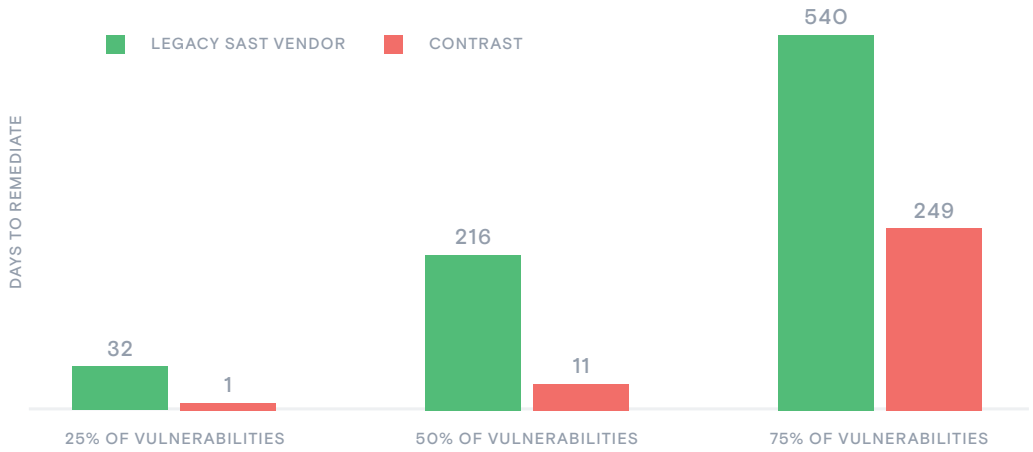


FIGURE 32

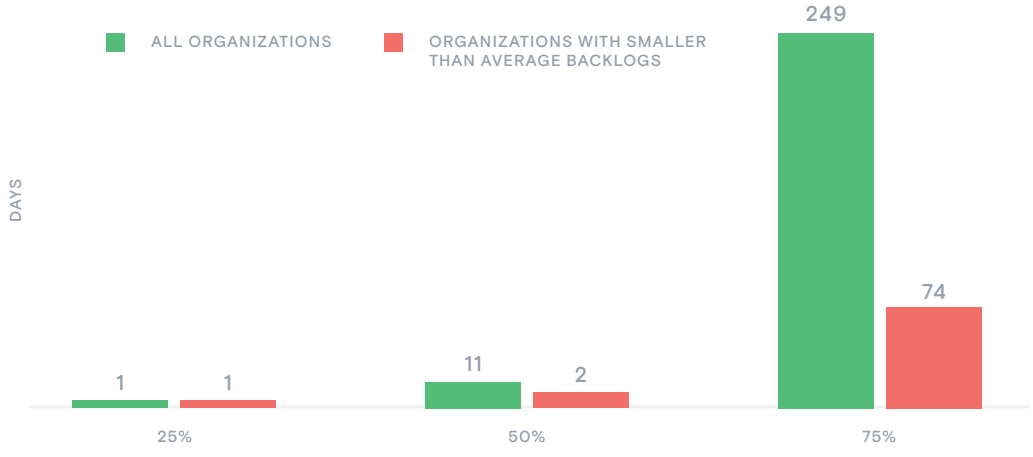


BY LEVEL OF SECURITY DEBT

Organizations with below-average security debt perform even better when it comes to remediation timelines. Their median time to remediate—the amount of time to close half of vulnerabilities identified—is two days, compared with 11 for all organizations (Figure 33). And they resolve 75% of vulnerabilities in 74 days versus 249. With a smaller backlog to worry about, they are able to focus more energy on resolving vulnerabilities as they occur.

FIGURE 33

Vulnerabilities Remediated

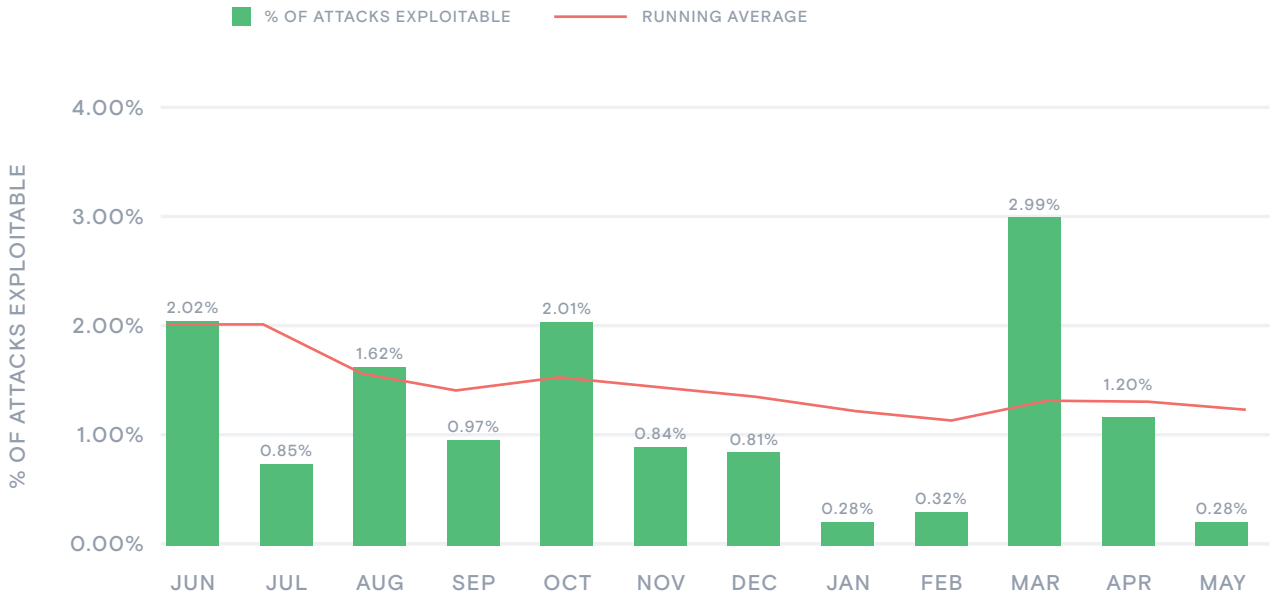


10 Attacks: Increased Prevalence, Decreased Viability

10 | Attacks: Increased Prevalence, Decreased Viability

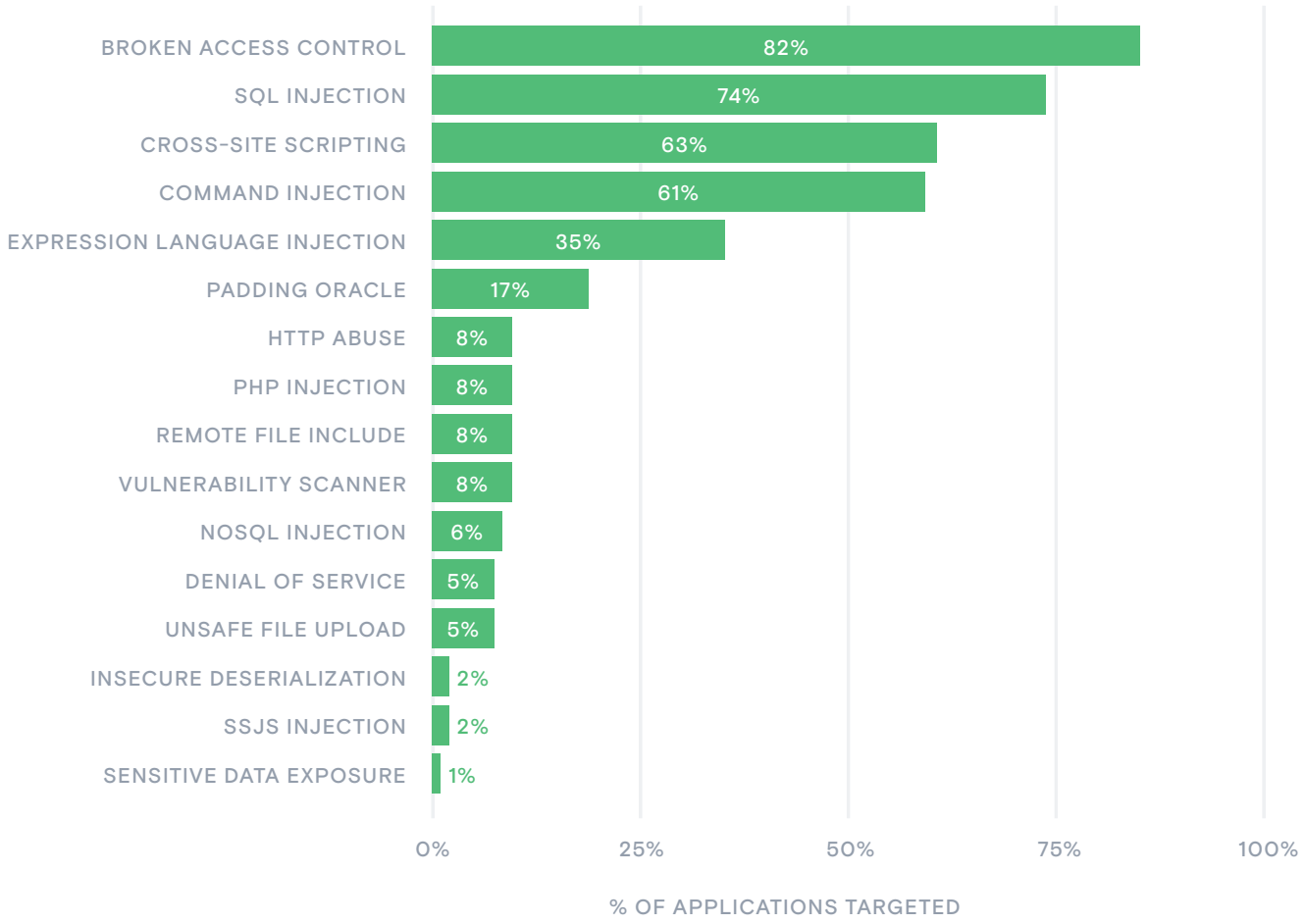
As with other types of cyberattacks, attacks on the application and API layers show no sign of letting up. The monthly average in our dataset for the past year was 4,841 attacks per application per month. And the typical organization was on the receiving end of 25,343 attacks per month.

FIGURE 34



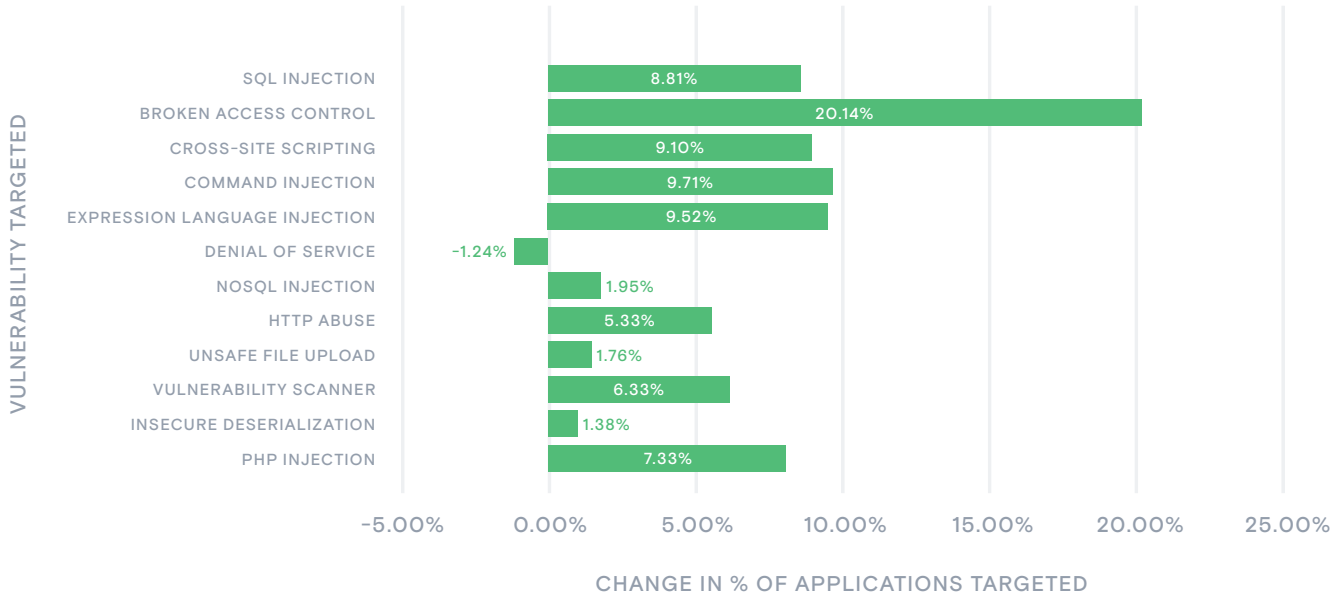
But all attacks are not created equal. This is because a vast majority of attacks are probes and do not hit an existing vulnerability. Just 1% of attacks were viable—that is, not probes—in the past 12 months. This is half the 2% of attacks that were viable the previous year (Figure 34). And while this appears as good news, it may mask other issues—specifically, probes can also help attackers narrow down the elements of an organization's attack surface that are most vulnerable.

FIGURE 35



Several attack types hit more applications this year than last year. SQL injection attacks impacted 74% of applications compared with 65% last year (Figure 35), a 13.8% increase. Broken access control rose even more—from 62% to 82%, a 32.3% increase. XSS, command injection, and EL injection were all up by 9 or more percentage points. In fact, 11 out of the top 12 attack types hit more applications this year than last year (Figure 36)—often with probes.

FIGURE 36



BY LANGUAGE

Not surprisingly, Java and .NET applications experienced somewhat different mixes of attack types. The increased prevalence of broken access control attacks was mostly driven by .NET applications, while EL injection attacks increased by 18 percentage points in Java applications (Figure 37).

FIGURE 37

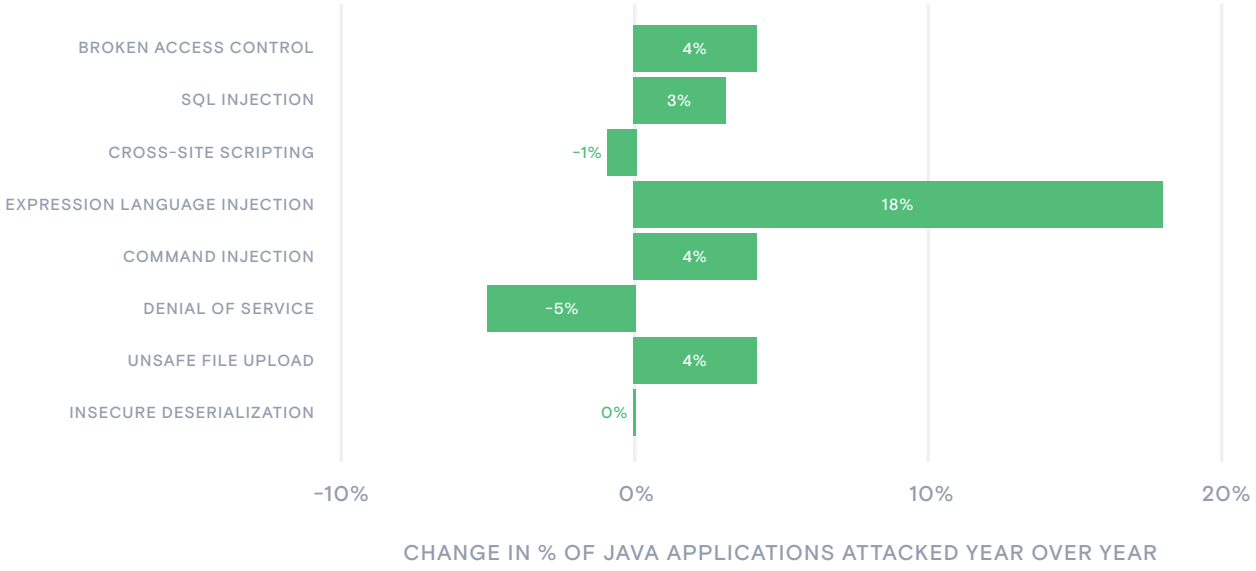
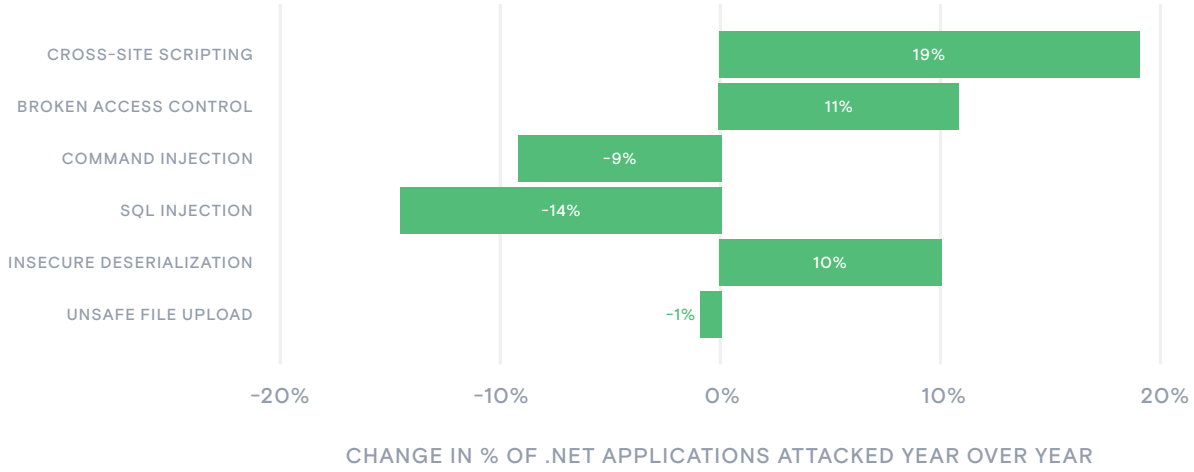


FIGURE 38


CROSS-SITE SCRIPTING
BROKEN ACCESS CONTROL
COMMAND INJECTION
SQL INJECTION
INSECURE DESERIALIZATION
UNSAFE FILE UPLOAD
19%
11%
-9%
-14%
10%
-1%
-20%
-10%
0%
10%
20%
CHANGE IN % OF .NET APPLICATIONS ATTACKED YEAR OVER YEAR

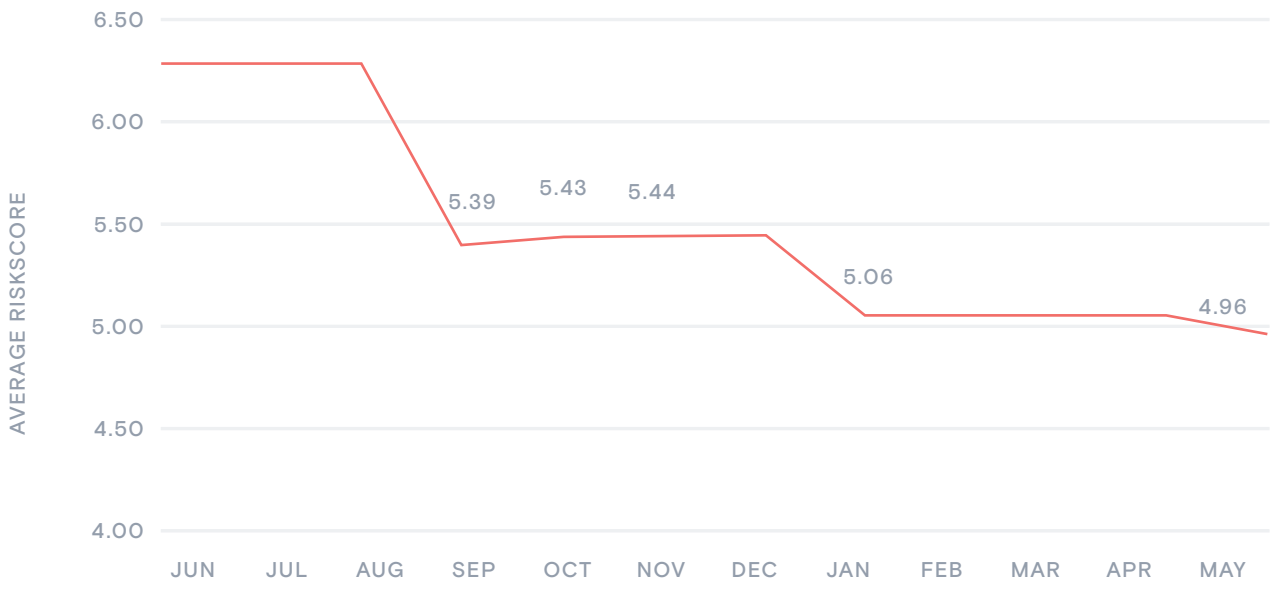
11 Contrast Riskscore™ Index: Overall, a Downward Trend

11 | Contrast Riskscore Index: Overall, A Downward Trend

We begin this report with a 12-month analysis of the Contrast RiskScore Index, which was introduced with the July–August 2020 bimonthly Application Security Intelligence Report. The index is a numerical score that provides an objective way to rank and visualize the relative risk presented by different vulnerability types over time. When beta version .5 of the index was finalized, scores were calculated retroactively to January 2020 for a special report that explains the index and analyzes trends in application risk for the 2020 calendar year.14 This is now planned to be a biannual report.

Perhaps the most important trend in the RiskScore Index since its inception is the steady decline in scores for the typical vulnerability type over this period. Specifically, the average RiskScore for the 19 vulnerability types we measure declined from 6.27 in June 2020 to 4.96 in May 2021 (Figure 39). This decline in overall risk can be attributed to the dataset we use in our calculations—applications protected by the Contrast Application Security Platform. As specific applications gain tenure with the platform and organizations accumulate experience in using it, overall risk declines as vulnerabilities are dealt with in a timely and efficient manner.

FIGURE 39

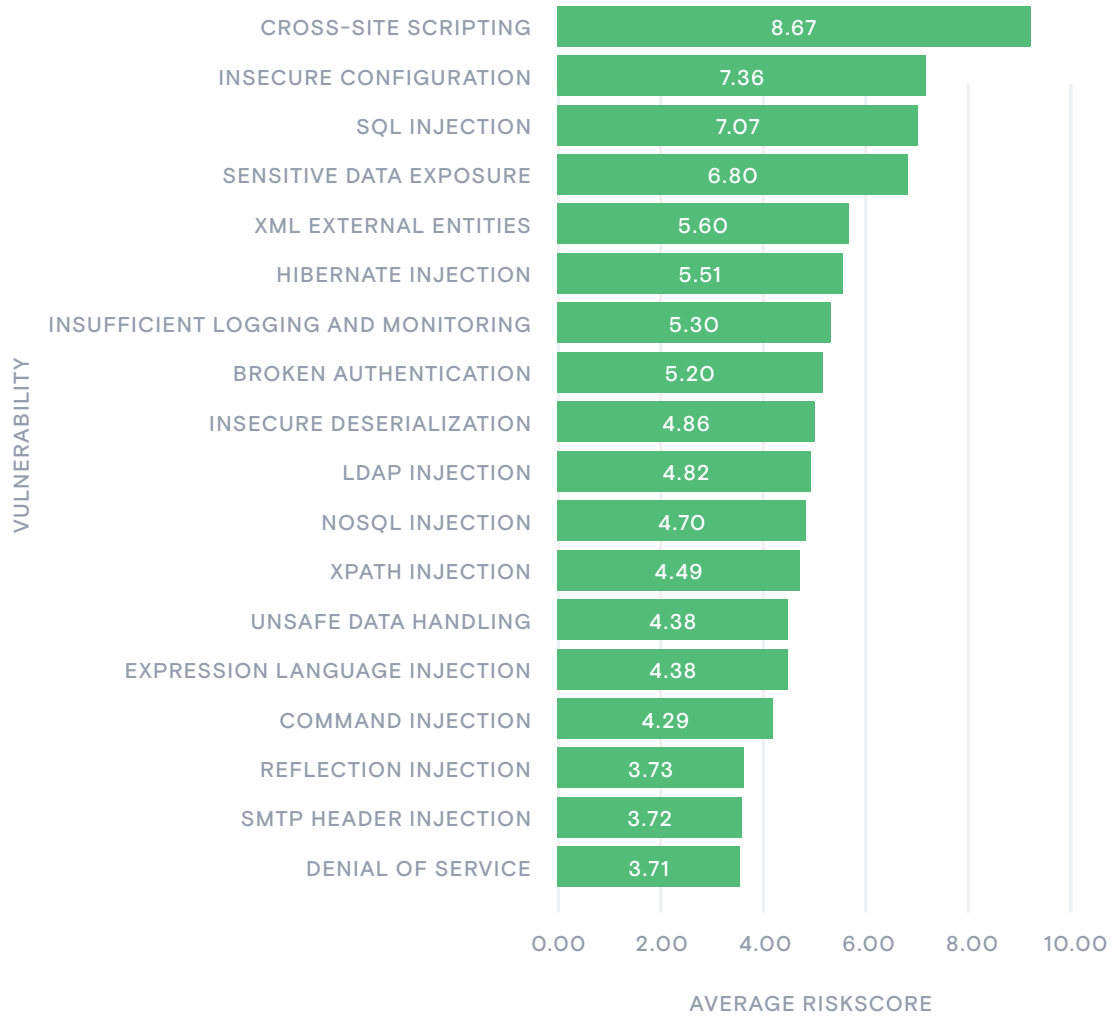


Overall, the five highest RiskScores have remained remarkably consistent since we have been calculating them (Figure 40). Broken access control and cross-site scripting (XSS) have had the first and second highest scores, respectively, for most of the past 12 months, and have been the only two vulnerability types scoring above 8 since November 2020 (Figure 41).

Insecure configuration, SQL injection, and sensitive data exposure round out the top five in terms of 12-month average scores, and each of them appeared in the top five in at least 10 out of the 12 months—albeit with reduced overall scores later in the year than earlier. SQL injection, in particular, saw a sharp decline in its RiskScore, falling from 8.87 in July 2020 to 5.35 in May 2021. But declines in other vulnerability types kept it in the top five almost every month.

Some mid-tier vulnerability types saw significantly more fluctuation over the period. As we have observed in several bimonthly reports, these fluctuations can inform security and development teams as they make short-term adjustments in prioritization based on the current risk landscape. Hibernate injection posed significant risk in the summer of 2020 with a score of 7.4 before steadily declining to its low of 3.83 in June. NoSQL injection scored above 7 that same summer before hitting a low of 2.7 in April of this year. Strikingly, EL injection declined from 6.43 to 2.68.

FIGURE 40

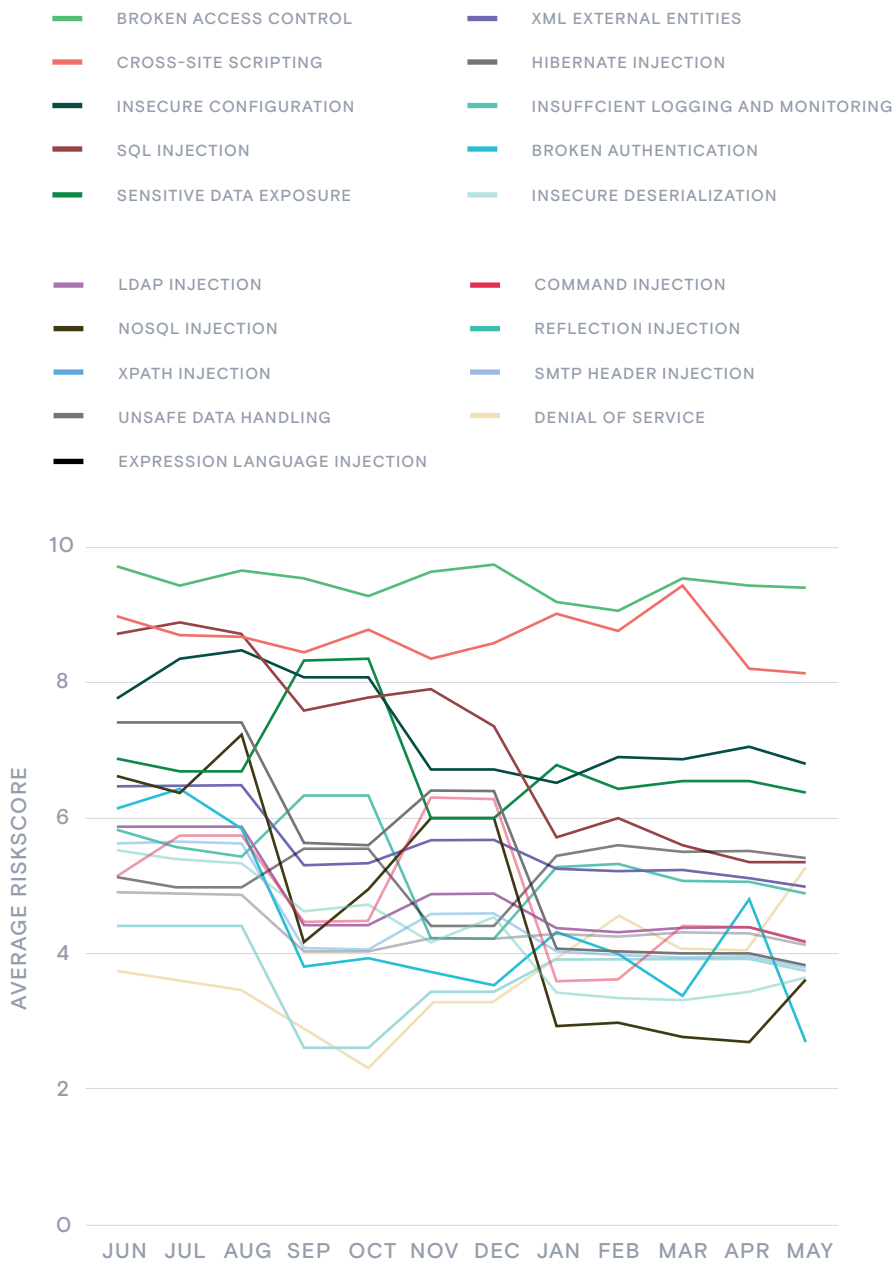


CONTRAST RISKSCORE INDEX

Contrast Labs research found that at nearly three-quarters of organizations, detection, investigation, remediation, and verification of fixes consume five or more hours of time per vulnerability for the development and security teams.[15] Given this, most organizations simply do not have the bandwidth to remediate all vulnerabilities and eliminate their security debt all at once, even as application security risks rise. This means that prioritization is key to reducing risk.

Unfortunately, organizations have lacked a tool to assess this risk in a dynamic way. While actuarial data is used to evaluate many other areas of risk, telemetry data of vulnerabilities and attacks is seldom used to assess application security risk, and existing tools and frameworks simply do not meet requirements of simplicity, scale, and visibility into current trends.injection scored above 7 that same summer before hitting a low of 2.7 in April of this year. Strikingly, EL injection declined from 6.43 to 2.68.

FIGURE 41



To make a contribution toward a solution, Contrast Labs leveraged its team's decades of experience to develop a numerical score based on the current risk posed by each vulnerability type.[16] The resulting RiskScore Index is simple to understand and use, dynamically adapts to real-world data, and allows for uncertainty and missing data.

Found in Contrast's bimonthly intelligence reports and the annual Observability Report, the currently published RiskScore provides a universal view based on telemetry from the entire Contrast user base. The ultimate goal is to release an open-source tool that will enable organizations to calculate their own RiskScores to quantify specific risk levels for vulnerabilities in their own applications or at the organizational level.

12 Conclusion

12 | Conclusion

The 2021 Application Security Observability Report leverages data from actual applications to bring insights into the composition of software, custom code and open-source vulnerabilities and attacks, and security debt to provide insight into how organizations can best protect themselves. The following key takeaways may be gleaned from this analysis:

- Less than 10% of an application's code is active third-party code, and only 20% is custom code. This means vulnerabilities present in nearly three-quarters of the typical application's code likely do not pose risk to an organization. Contrary to perceptions, 22% of active code is from libraries and frameworks, and custom code accounts for a large majority of vulnerability risk. Without visibility into which libraries and library classes are active, organizations waste time on non-risky vulnerabilities, often at the expense of more risky ones. This shines a light on the need for the observability required to effectively prioritize remediation.

- The likelihood of serious vulnerabilities is increasing. While the sheer number of vulnerabilities continues to be high, the biggest worry is that a higher proportion of those vulnerabilities is serious. The percentage of applications containing at least one serious vulnerability increased by nearly 31%. This means organizations need robust observability across the SDLC to effectively prioritize remediation and response.

- Route coverage hits 80% in the first year in most applications. Contrast provides a unique view of application security by "route" that is organized by exposed attack surface. On average, over 84% of all application routes were exercised at least once, and 57% are exercised each month. Over the past year, Contrast found that 68% of applications secured with Contrast Assess achieved 80% coverage in 12 months. Further, the report discovered that both exercised and unexercised routes pose risk: an average 82% of applications expose routes with vulnerabilities.

- Organizations' security debt per application declined as complexity increased. The Contrast customers in the dataset have just 21 backlogged vulnerabilities per application, compared with 26 last year. However, a proliferation of new, increasingly complex applications means that the typical organization still has more than 1,900 unaddressed vulnerabilities companywide, a reminder that decreasing security debt is a long process.

- Vulnerability prevalence for Contrast customers shrinks dramatically following deployment, which is described as vulnerability escape rate (VER). An average of 6 new serious vulnerabilities and 12 new vulnerabilities in general are introduced into applications during the first two months of Contrast deployment. This VER is cut in half to 3 and 6, respectively, after nine months and reduced almost altogether to 0 and 1, respectively, after one year.

- Contrast users continue to experience dramatically faster median time to remediate over traditional technology. The milestone of closing 50% of identified vulnerabilities occurs much earlier for the Contrast customers in our dataset than the data reported by a traditional SAST vendor. When all vulnerabilities are taken into account, the median time to remediation is 11 days compared with 216 days with the traditional approach. This is because application security testing is continuous, more evenly spread across the SDLC, and actionable feedback to testing is immediate. Companies with below-average security debt fare even better, achieving a median time to remediation of 2 days.

- While the volume of attacks continues to rise, much of this activity amounts to noise, and the percentage of attacks that were viable declined by half compared with last year. However, even with around 1% of attacks viable, they still connect with hundreds of vulnerabilities each month that can be exploited.

As applications become more critical for the business, application security must become a bigger focus in an organization's overall cybersecurity efforts. It is critical that companies take a comprehensive approach to delivering secure software and protecting it once released. Detailed observability on vulnerabilities, attacks, and open-source usage and security enables organizations to succeed at these two essential goals, and much of this data is only provided by tools from Contrast Security. In an increasingly digital marketplace, this kind of visibility is no longer an option.

13 Report Contributors

13 | Report Contributors

JEFF WILLIAMS

CTO AND CO-FOUNDER, CONTRAST SECURITY

Jeff brings more than 20 years of security leadership experience as Co-Founder and Chief Technology Officer of Contrast. Previously, Jeff was Co-Founder and Chief Executive Officer of Aspect Security, a successful and innovative application security consulting company acquired by Ernst & Young. Jeff is also a founder and major contributor to OWASP, where he served as Global Chairman for eight years and created the OWASP Top 10, OWASP Enterprise Security API, OWASP Application Security Verification Standard, XSS Prevention Cheat Sheet, and many other widely adopted free and open projects. Jeff has a BA from the University of Virginia, an MA from George Mason, and a JD from Georgetown.

DAVID LINDNER

CHIEF INFORMATION SECURITY OFFICER, CONTRAST SECURITY

David is an experienced application security professional with over 20 years in cybersecurity. In addition to serving as the chief information security officer, David leads the Contrast Labs team that is focused on analyzing threat intelligence to help enterprise clients develop more proactive approaches to their application security programs. Throughout his career, David has worked within multiple disciplines in the security field—from application development, to network architecture design and support, to IT security and consulting, to security training, to application security. Over the past decade, David has specialized in all things related to mobile applications and securing them. He has worked with many clients across industry sectors, including financial, government, automobile, healthcare, and retail. David is an active participant in numerous bug bounty programs.

BRIAN GLAS

ASSISTANT PROFESSOR OF COMPUTER SCIENCE, UNION UNIVERSITY

Brian possesses nearly 20 years of experience in various roles in IT and over a decade in application development and security. In addition to teaching a full load of classes at Union University, Brian serves as a part-time management consultant and advisor for Contrast Labs. He worked on the Trustworthy Computing team at Microsoft and served as a project lead and active contributor for SAMM v1.1-2.0 and OWASP Top 10 2017. He is a popular speaker at numerous conferences and online events, having presented at InfoSec World, Cloud Security World, and numerous OWASP conferences and meetings. Brian is also an author of various papers and is currently researching writing a book on application security. He holds a long list of cybersecurity and IT certifications as well as a master in business administration and bachelors in computer science from Union University.

Contrast SECURITY

KATHARINE WATSON

DATA SCIENTIST, CONTRAST SECURITY

Katharine is a driving force in developing and building data analytics frameworks for Contrast—including Contrast Labs—and turning data into actionable narratives and insights for internal and external customers. Katharine worked as an analyst, consultant, and project manager in both private and nonprofit organizations. Before launching a career in data science. Katharine worked for three years as a mathematics teacher in the Teach for America program. Katharine holds undergraduate and graduate degrees from The Johns Hopkins University.

PATRICK SPENCER, PH.D.

EDITOR IN CHIEF, INSIDE APPSEC PODCAST
HEAD OF CONTENT AND PR/COMMUNICATIONS, CONTRAST SECURITY

Patrick founded and serves as the editor in chief for the Inside Appsec podcast and leads the content marketing and PR/communications team at Contrast. He has more than a decade and a half of experience in various senior marketing and research roles within the cybersecurity sector and is the recipient of numerous corporate and industry awards. After leaving the corporate world to start his own agency, Patrick joined Fortinet to lead content marketing and research. His many duties included serving as the editor in chief for The CISO Collective. Patrick's roots in cybersecurity go back to Symantec, where he spent nearly a decade in senior marketing roles of increasing scope and responsibility. While at Symantec, Patrick served as the editor in chief for CIO Digest, an award-winning digital and print publication containing strategies and insights for the technology executive.

MARK MULLINS

FOUNDER AND PRINCIPAL, MHM CONTENTSOURCE

MHM ContentSource specializes in marketing research and writing projects for clients across the technology sector. Mark has 15 years of experience in research and content marketing across the technology sector, as both an employee and a consultant. He has authored numerous research reports, white papers, and magazine features and produced dozens of marketing videos and a podcast series. His work has been published by leading technology brands such as Symantec, LivePerson, PRO Unlimited, Finastra, Fortinet, Lastline, and Contrast Security, among others.

[1] "How COVID-19 has pushed companies over the technology tipping point—and transformed business forever," McKinsey, October 5, 2020.
[2] "2021 Data Breach Investigations Report," Verizon, May 2021.
[3] "2021 State of Application Security in Financial Services Report," Contrast Security, May 2021.
[4] "Priorities and Challenges for Modern Software Developers," Contrast Security, March 2021.
[5] "2021 State of Open-source Security Report," Contrast Security, April 2021.
[6] "2021 Data Breach Investigations Report," Verizon, May 2021, found that 39% of data breaches were the result of application vulnerabilities.
[7] Michael Riley, et. al, "Russia-Linked SolarWinds Hack Snags Widening List of Victims," Bloomberg, December 17, 2020.
[8] "Executive Order on Improving the Nation's Cybersecurity," The White House, May 12, 2021.
[9] "2020 Open Source Security and Risk Analysis Report," Synopsys, May 2020.
[10] Liam Tung, "Open-source security: This is why bugs in open-source software have hit a record high," ZDNet, March 13, 2020.
[11] "2021 State of Open-source Security Report," Contrast Security, April 2021.
[12] "A Basic Timeline of the Exchange Mass-Hack," Krebs on Security, March 8, 2021; Charlie Osborne,
"Updated Kaseya ransomware attack FAQ: What we know now," ZDNet, July 23, 2021.
[13] "State of Software Security v11," Veracode, accessed July 23, 2021.
[14] "Prioritizing Application Security Risk Management With the Contrast RiskScore," Contrast Security, June 2021.
[15] "The State of DevSecOps Report," Contrast Security, November 2020.
[16] "Prioritizing Application Security Risk Management With the Contrast RiskScore," Contrast Security, June 2021.

Contrast Security provides the industry's most modern and comprehensive Application Security Platform, removing security roadblocks inefficiencies and empowering enterprises to write and release secure application code faster. Embedding code analysis and attack prevention directly into software with instrumentation, the Contrast platform automatically detects vulnerabilities while developers write code, eliminates false positives, and provides context-specific how-to-fix guidance for easy and fast vulnerability remediation. Doing so enables application and development teams to collaborate more effectively and to innovate faster while accelerating digital transformation initiatives. This is why a growing number of the world's largest private and public sector organizations rely on Contrast to secure their applications in development and extend protection in production.

240 3rd Street
2nd Floor
Los Altos, CA 94022
Phone: 888.371.1333
Fax: 650.397.4133

Contrast SECURITY

contrastsecurity.com