CONTRAST SECURITY

2020 APPLICATION SECURITY OBSERVABILITY REPORT

Connecting Vulnerability and Threat Analysis with Real-world Implications in Applications and APIs

2020 REPORT

TABLE OF CONTENTS

01 | FOREWORD

The importance of digital transformation has never been more important than it is today. Competitive differentiation is tethered to an organization's ability to deliver new customer experiences, tap new revenue opportunities, lower costs, and improve efficiencies through the development of new applications.

Contrast Labs is very excited to share its research findings over the past year with the intent that they will guide organizations in transforming their businesses through digital innovation while ensuring application security. More applications, more application programming interfaces (APIs), more connections and transactions, and more data create a broader application attack surface—and this has not gone unnoticed by cyber criminals. Nearly one-half of data breaches this past year were the result of application vulnerabilities—more than double the previous year.

You will notice that this year's report is broken into four main sections—vulnerabilities, attacks, median and mean time to remediate, and open-source dependencies and risks. You can read these individually or in aggregate. The report also includes an analytical breakdown across industry segments and languages that provide additional direction and insights. The close of the report aggregates the data into key takeaways for development, security, and operations leaders.

Not every vulnerability should be treated the same. Upwards of 96% of applications contain at least one vulnerability, yet only 26% of them have a serious vulnerability. It is impossible to fix all vulnerabilities at once, and organizations must prioritize vulnerability remediation to effectively and efficiently manage application risk.

At the same time, with individual applications experiencing over 13,000 attacks each month, organizations must ensure the same protections they have in place for development are extended to software in production. Data shows that 98% of these attacks are probes and do not hit an existing vulnerability. But even with just 2% reaching a vulnerability, this poses serious risk and requires a concerted security approach from development and into production.

Security debt (viz., the accumulation of unresolved vulnerabilities) certainly plays an important role in defining risk. Report data shows that those with below-average security debt have a significantly better risk posture than those organizations that allow vulnerabilities to fester. Indeed, vulnerabilities that age beyond 90 days have a greater likelihood of going unresolved for more than a year.

Without open-source frameworks and libraries, modern software development simply would be unable to meet the aggressive velocity and agility demands businesses now expect. But more than half of open-source libraries are inactive. Once again, this level of observability is crucial in helping organizations determine which libraries matter—and do not—and how they prioritize version updates.

We designed our 2020 report with the intent of enabling organizations to fully embrace the potential modern software offers to all industries and business units. And by making security observable, we believe development, security, and operations teams will be empowered to collaborate—achieving the high velocity and automated software assurance required in today's digital world.

Sincerely,

JEFF WILLIAMS
CTO AND CO-FOUNDER

DAVID LINDNER
DIRECTOR OF APPLICATION SECURITY

02 | EXECUTIVE SUMMARY

Contrast's "2020 Application Security Observability Report" provides insights gleaned from analysis of aggregate telemetry generated from applications during development, testing, and operations from Contrast Security customers between June 2019 and May 2020. Key findings include:

- **Vulnerabilities.** Nearly all applications have at least one vulnerability, and more than one-quarter have a serious one. 11% of applications have more than six serious vulnerabilities. Well over half of applications have insecure configuration and sensitive data exposure vulnerabilities. Notably, twice as many Java applications have at least one serious vulnerability than .NET ones.

- **Time to remediation.** Contrast customers achieved a median time to remediate of seven days as compared to 121 days for customers of one static application security testing (SAST) vendor. The differences are even more dramatic when serious vulnerabilities are examined, with 25% being remediated in one day and 75% in 16 days. This faster remediation time translates into both lower risk and security debt—with Contrast customers achieving a median time to remediation of just one day. At the same time, customers with below-average security debt (viz., fewer vulnerabilities) see a 1.7x better risk posture than all customers.

- **Open-source libraries.** The average application has content from 32 different libraries, though only 45% of those libraries are actually used by the application. The top Common Vulnerabilities and Exposures (CVEs) for software written in Java have significantly higher Common Vulnerability Scoring System (CVSS) scores than the CVSS scores for the top .NET CVEs, suggesting higher risk for Java applications. Organizations should manage open-source libraries in such a way that the versions they use do not put them at risk, as the use of older versions can result in increased security debt.

- **Attacks.** On average, each application endured more than 13,000 attacks per month in the past year, with injection, cross-site scripting, and broken access control topping the attack-vector list. Fortunately, 98% of attacks do not hit an existing vulnerability. The high volume of attempts to infiltrate applications accentuates the need to effectively prioritize remediations and take steps to block attacks on applications in production. Organizations can protect themselves by taking a strategic, risk management-based approach to application security. This means prioritizing vulnerabilities according to the risk they pose, which requires organizations to have actionable data not only at an industry level but also for the specific organization.

KEY FINDINGS

96% of applications have at least **1 VULNERABILITY**

26% have at least **1 SERIOUS VULNERABILITY**

11% have at least **6 SERIOUS VULNERABILITIES**

MOST COMMON SERIOUS VULNERABILITIES:

CROSS-SITE SCRIPTING — **15%**

BROKEN ACCESS CONTROL — **13%**

SQL INJECTION — **6%**

THE AVERAGE APPLICATION CONTAINS CODE FROM 32 DIFFERENT LIBRARIES BUT ONLY 14 ARE ACTIVE

86%+ OF OPEN-SOURCE LIBRARY USES DO NOT USE THE LATEST VERSION

MEDIAN TIME TO REMEDIATION FOR CONTRAST CUSTOMERS: 7 DAYS COMPARED WITH 121 DAYS WITH SAST

25% OF SERIOUS VULNERABILITIES REMEDIATED IN 1 DAY—COMPARED WITH 19 DAYS WITH SAST

MEAN TIME TO REMEDIATION: 67 DAYS (32 DAYS FOR ORGANIZATIONS WITH BELOW-AVERAGE SECURITY DEBT)

13,279 AVERAGE ATTACKS PER APPLICATION, PER MONTH

60%+ GROWTH RATE FOR THE TOP FIVE VULNERABILITY ATTACK VECTORS OVER THE PAST YEAR

COMMAND INJECTION ATTACKS INCREASED BY 179%

03 | INTRODUCTION

Contrast's "2020 Application Security Observability Report" analyzes application security (AppSec) trends for the past 12 months ending on May 31, 2020. The goal of the research, which was performed by Contrast Labs, is to provide readers with a comprehensive view of the state of application vulnerabilities and attacks, time to remediation, and usage of open-source code. The dataset includes vulnerabilities identified by Contrast Assess, attacks detected by Contrast Protect, and open-source library data gathered by Contrast OSS.

This report summarizes application vulnerability and attack trends over the past year and includes actionable insights for security, development, and operations leaders and practitioners. The data is also broken down by industry and programming language. Contrast Security is the ***only player*** in the industry that provides perspective across the application life cycle by analyzing security vulnerabilities, library issues, and attacks in a single report, with bimonthly updates published to complement this annual report and provide security, development, and operations teams with regular trends and insights.

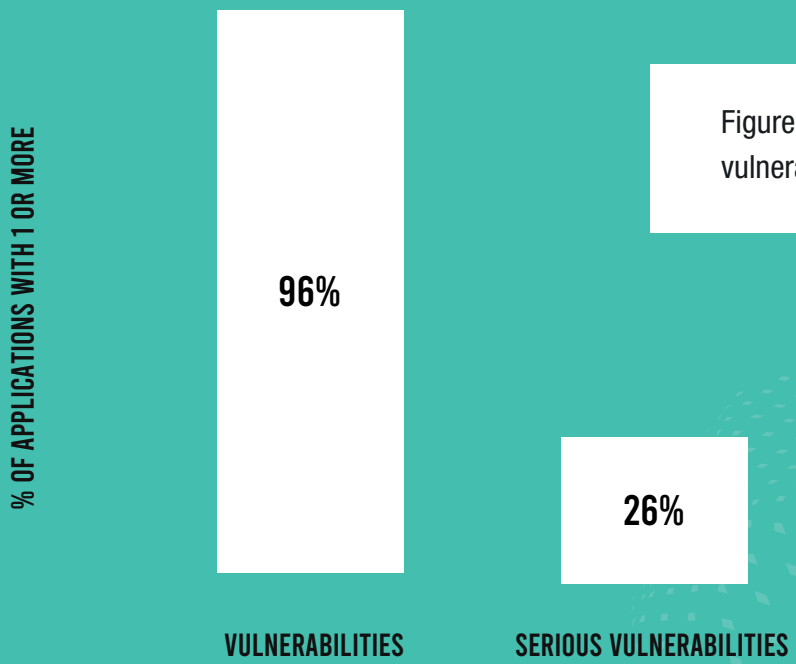


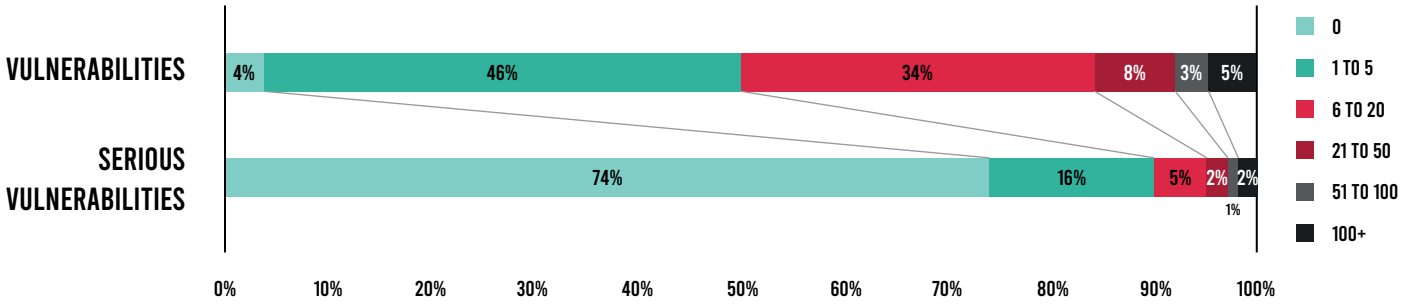
Figure 1. Percent of applications with reported vulnerabilities and serious vulnerabilities.

04

OVERALL VULNERABILITIES: STILL ALL TOO COMMON

04 | OVERALL VULNERABILITIES: STILL ALL TOO COMMON

Contrast Labs data confirms that an overwhelming majority of applications have vulnerabilities at some point in the software development life cycle (SDLC). Overall, 96% of applications have at least one ***serious*** vulnerability (see sidebar: "Ranking the Severity of Vulnerabilities," page 13), and 11% have at least six ***serious*** vulnerabilities (Figures 1 and 2).



% OF APPLICATIONS BY NUMBER OF VULNERABILITIES

Figure 2. Percent of applications by number of reported vulnerabilities and serious vulnerabilities.

BY VULNERABILITY CATEGORY

Well over half of applications have vulnerabilities in two categories (Figure 3): insecure configuration (72%) and sensitive data exposure (64%). Among serious vulnerabilities, the most common categories are cross-site scripting (XSS; 15%), broken access control (13%), and SQL injection (6%). It should be noted that almost all XSS and injection vulnerabilities that occur are serious.

Analyzing the frequency of vulnerabilities, Contrast Labs found that the most common serious vulnerabilities—XSS, broken access control, and injection—are most likely to occur multiple times in

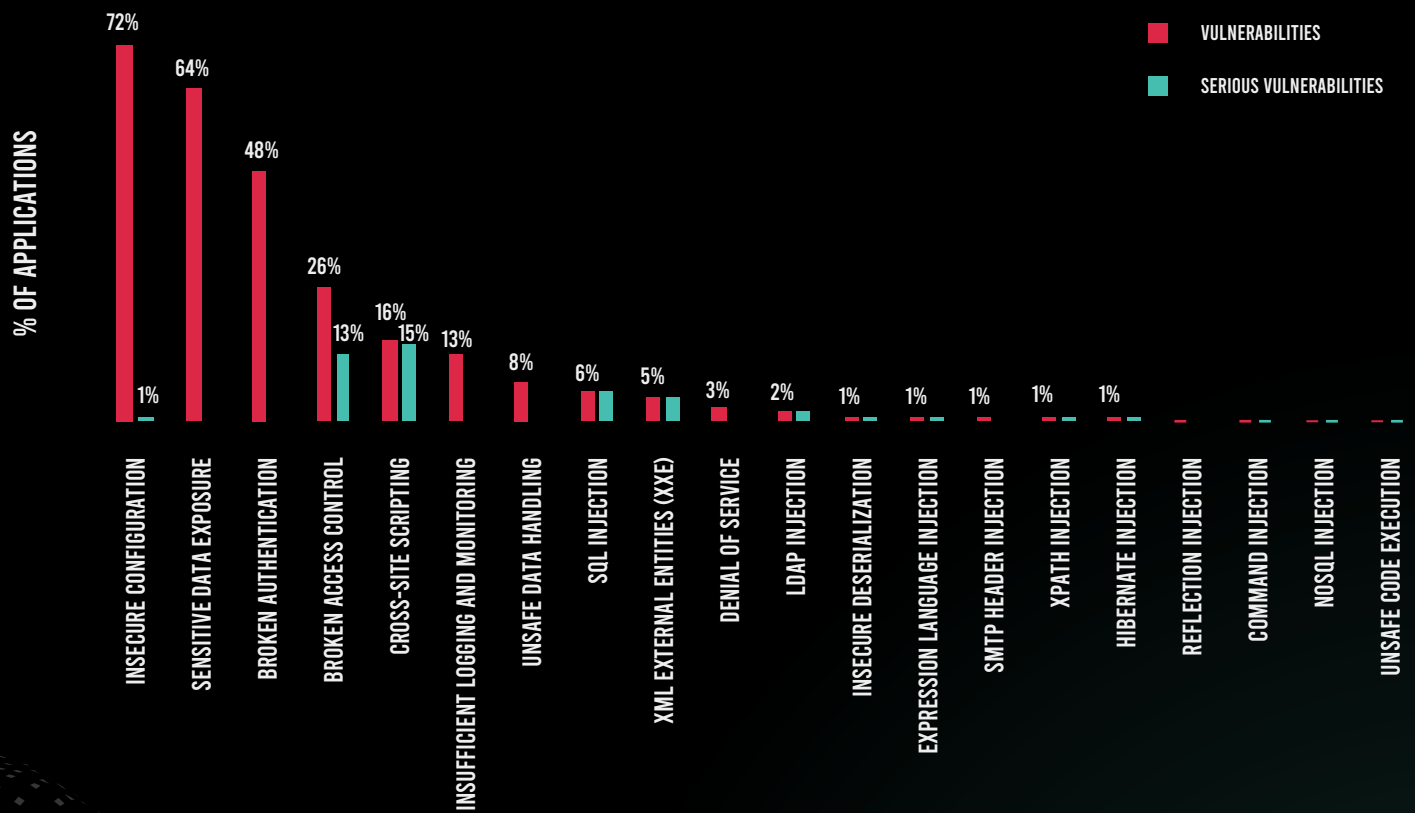


Figure 3. Percent of applications with reported vulnerabilities and serious vulnerabilities by specific vulnerability category.

vulnerable applications. Applications that have XSS and SQL injection vulnerabilities had a median of more than three reported instances of vulnerabilities in that category. In other words, half had at least that many occurrences. For broken access control, applications had a median of two instances—60% fewer vulnerability instances on average than expression language injection, which has a median of five instances.

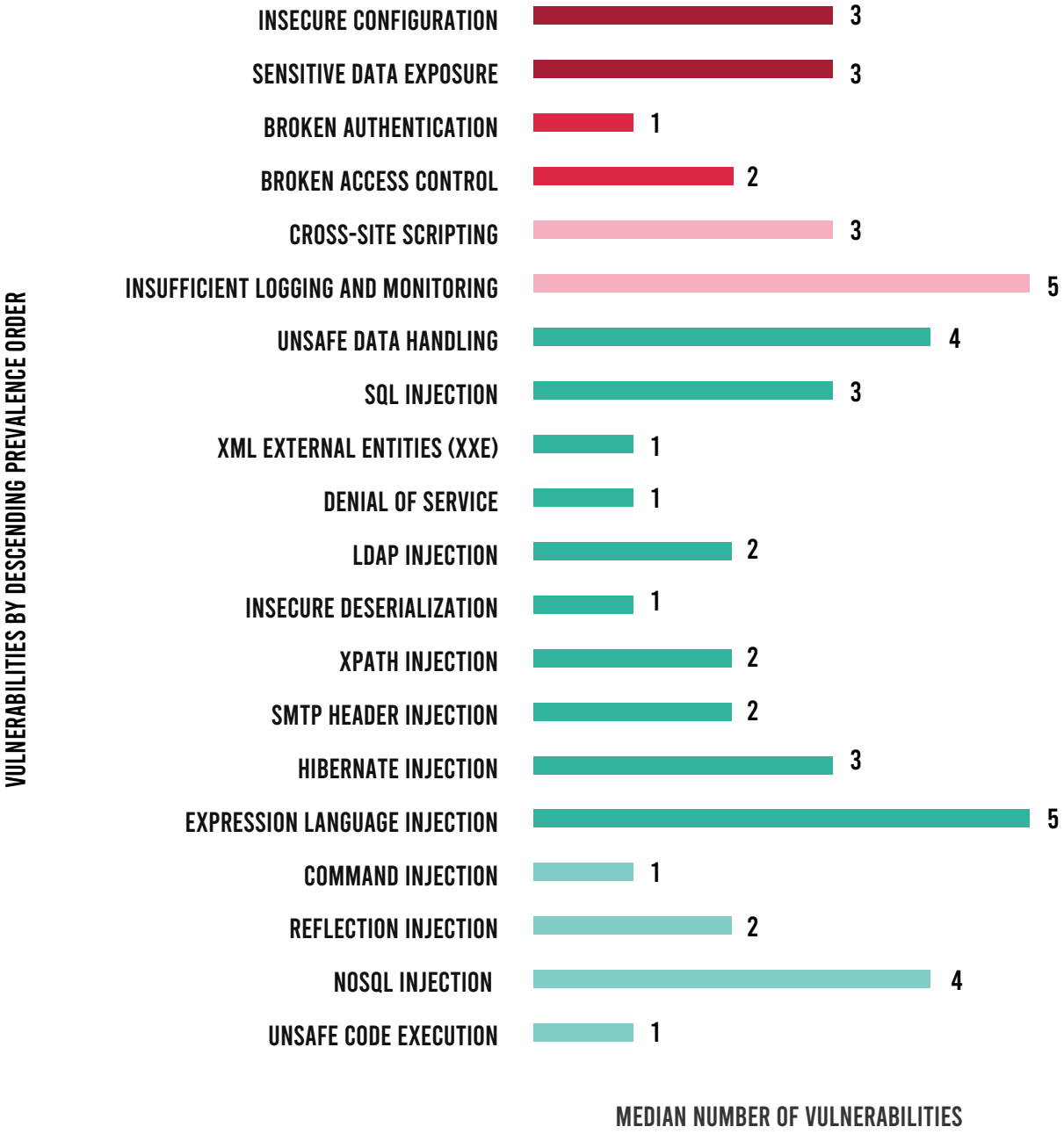


Figure 4. Median number of vulnerabilities per application in descending order of vulnerability prevalence.

On average, applications only reported vulnerabilities in three distinct categories and two distinct serious risk categories. Depending on the specific vulnerability type, a smaller number of categories could be good news for developers, as a single code change can potentially address multiple instances of a vulnerability. For instance, many configuration vulnerabilities can be fixed with code changes in a small number of places. As an example, vulnerabilities related to injection types can also be fixed with

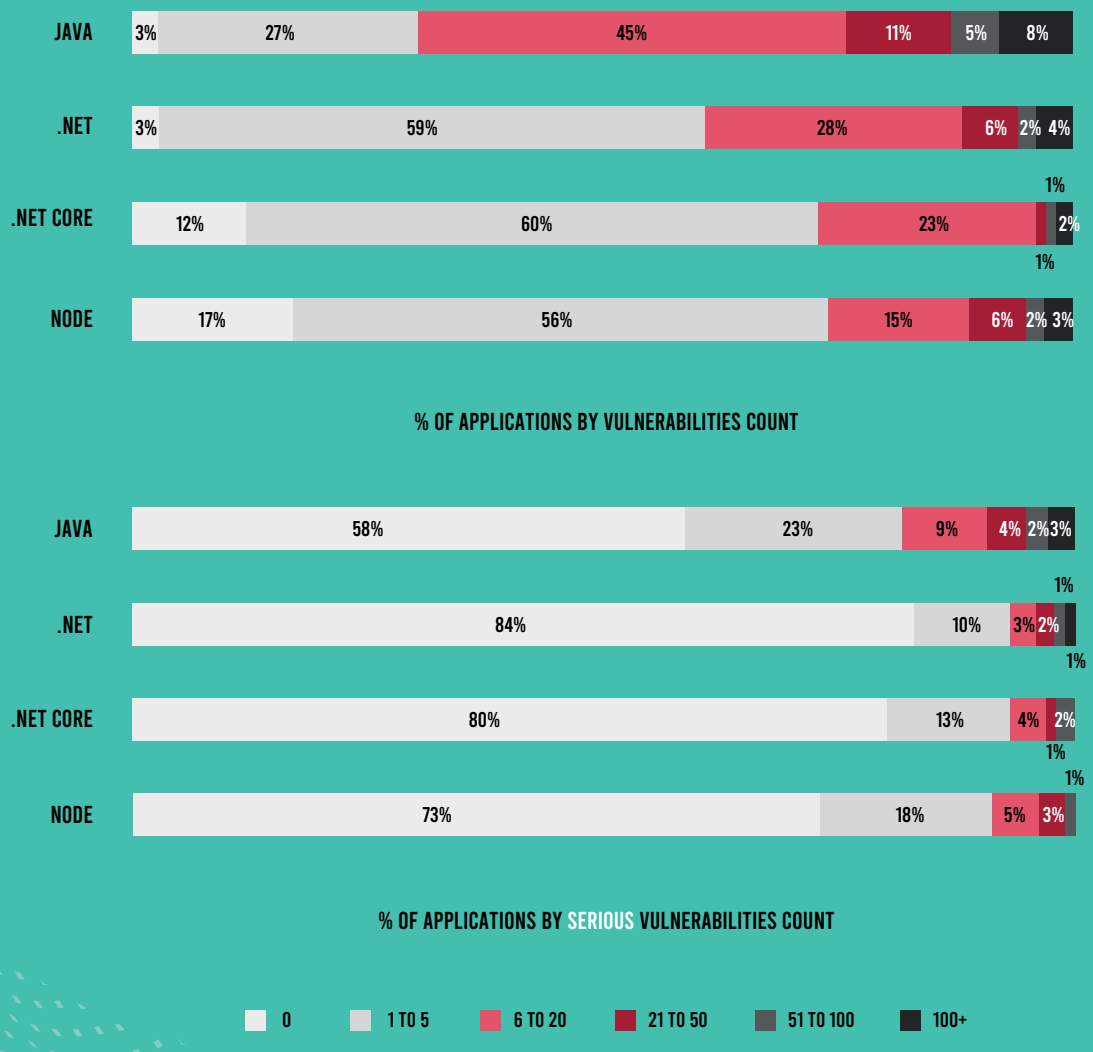


Figure 5. Percent of applications by number of reported vulnerabilities and serious vulnerabilities, by application language.

adjustments to only one or two locations, as there could be natural or architected bottlenecks that result in data flows all being processed by a common piece of code. In addition, identifying multiple vulnerabilities in a category can help security teams as they continuously improve the baseline for security testing.

BY LANGUAGE

While it is increasingly common for different parts of an application to be written in different languages, it is worth noting that Java applications are more likely to contain serious vulnerabilities than other languages in the dataset, most notably .NET. Specifically, over twice as many Java applications have at least one serious vulnerability compared to .NET and .NET Core—42% versus 16% and 20%, respectively (Figure 5). And 18% of Java applications have at least six serious vulnerabilities, while only 7% of .NET ones have that many. Particular problem areas for Java applications include broken access control (26%) and XSS (22%; Figure 6). This can be traced to a lack of standardization in Java, which is an open-source language—compared with .NET, which is highly standardized and controlled by Microsoft.

SIDEBAR: RANKING THE SEVERITY OF VULNERABILITIES

Open Web Application Security Project (OWASP) is a worldwide nonprofit focused on improving the security of software. The OWASP Top 10 is a document that reflects a broad consensus of vulnerability types in web applications that represent the most critical security risks for organizations. The latest version was released in 2017, with the next version scheduled in 2021. OWASP states that "[c]ompanies should adopt this document and start the process of ensuring that their web applications minimize these risks,"[1] but does not rank them according to the risk they pose.

Contrast Labs goes a step further by independently evaluating the severity of vulnerabilities by assigning them an impact rating (how damaging would it be if it were exploited) and a likelihood rating (what are the chances of it being exploited). The default rankings are built into the Contrast Agent's ruleset, and organizations can adjust these rules to fit their needs. In this report, vulnerabilities identified as serious are rated as either High or Critical, representing 28% of all vulnerabilities (Figure 7). Notably, 42% of vulnerabilities are rated as Medium in both likelihood and impact. For organizations that prioritize remediation according to risk, these vulnerabilities provide a clearly delineated second tier.

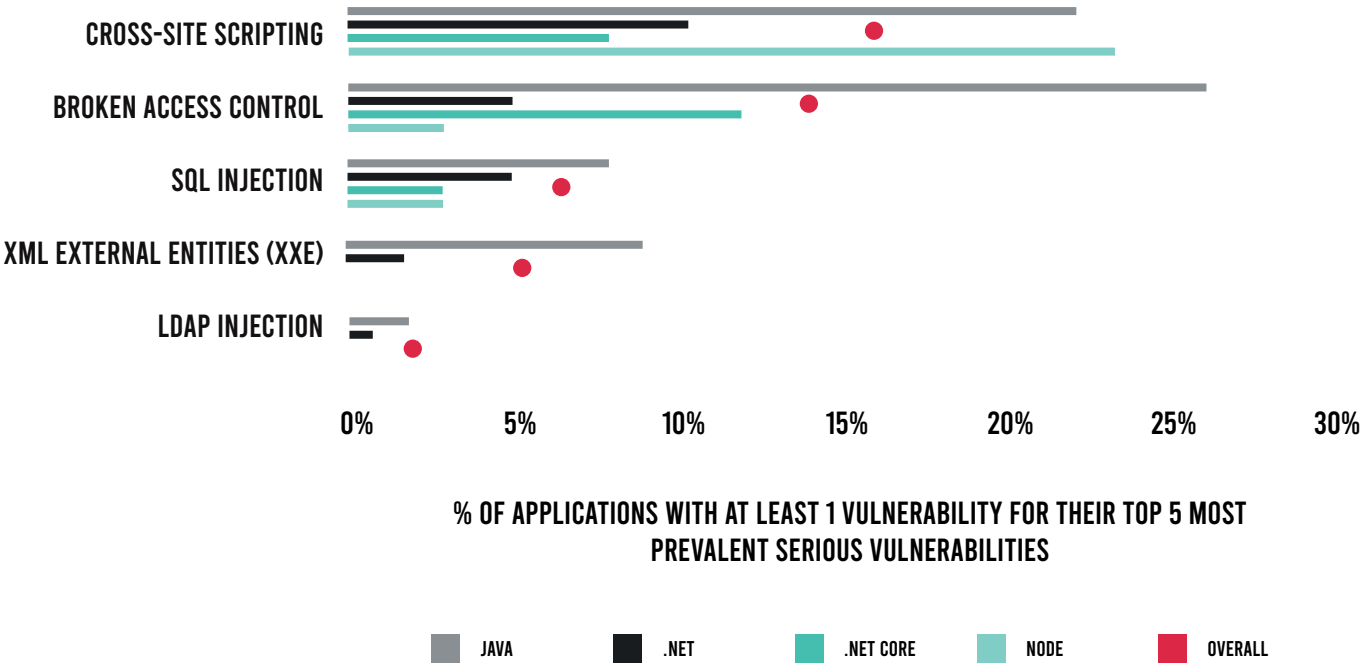


Figure 6. Percent of applications with reported serious vulnerabilities by application language and overall.

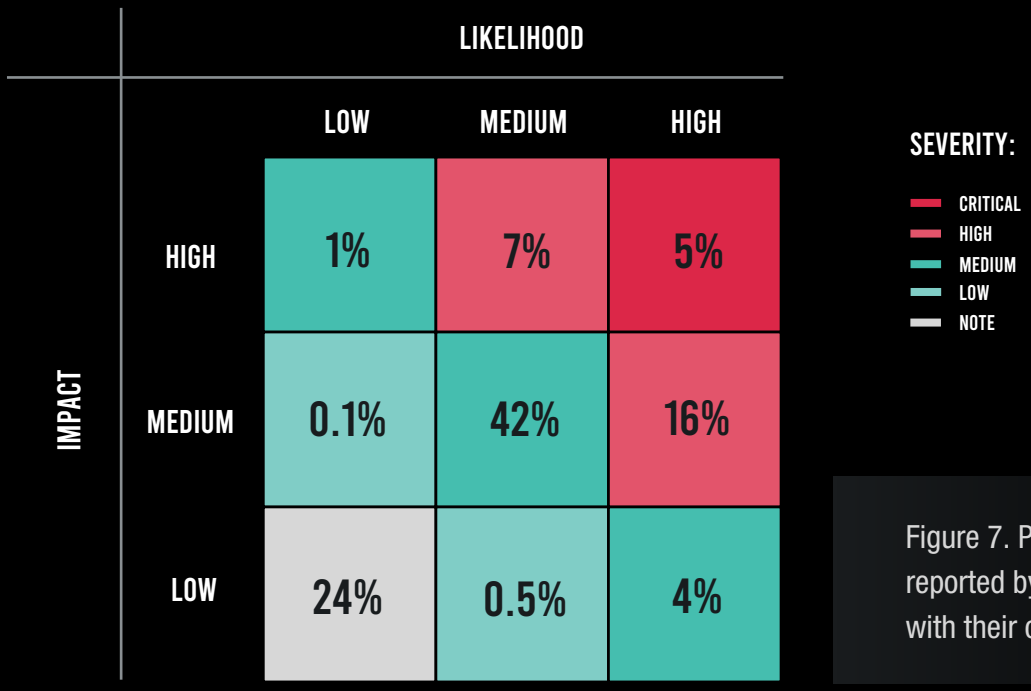


Figure 7. Percent of vulnerabilities reported by likelihood and impact, with their overall severity score.

05

VULNERABILITIES BY INDUSTRY: VULNERABILITIES ARE VERTICALLY AGNOSTIC

05 | VULNERABILITIES BY INDUSTRY: VULNERABILITIES ARE VERTICALLY AGNOSTIC

Contrast Labs also analyzed vulnerability data by industry. It could be argued that all companies are now software companies, but organizations in different industries deliver applications for vastly different purposes.[2] As a result, it might not be surprising for vulnerability data to vary greatly from vertical to vertical.

On the other hand, all developers use similar languages and tools, and analysis of Contrast Labs data suggests that vulnerability trends are remarkably uniform across the board. The percentage of applications with at least one serious vulnerability ranged from a high of 36% in government to a low of 19% in education (Figure 8).

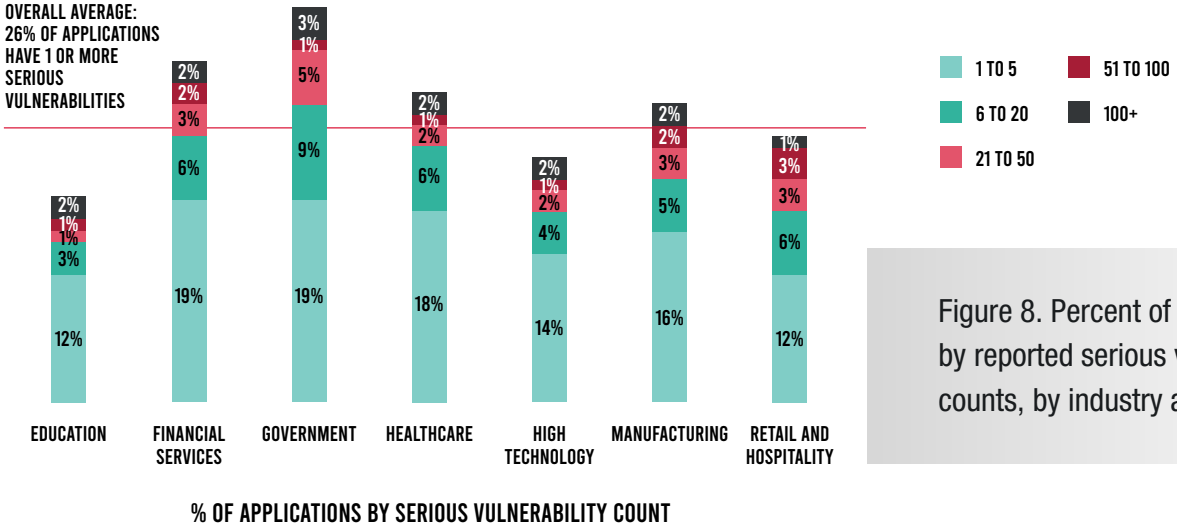


Figure 8. Percent of applications by reported serious vulnerability counts, by industry and overall.

Most industries have a small but significant subset of applications with a large proportion of overall vulnerabilities. In every vertical except high technology and education, more than 10% of applications had six or more serious vulnerabilities. XSS and broken access control are the two most common serious vulnerabilities in every industry (Figure 9).

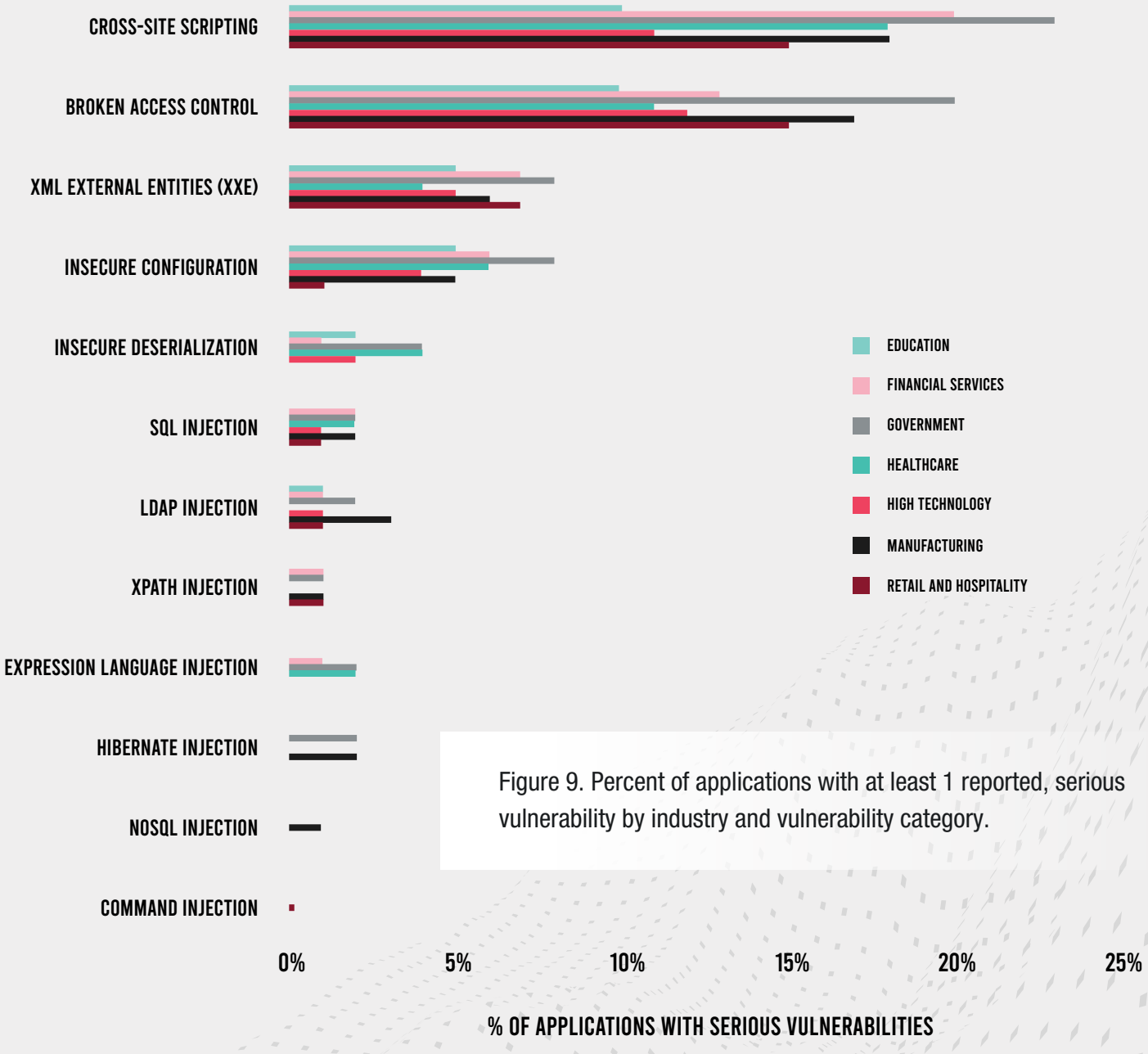


Figure 9. Percent of applications with at least 1 reported, serious vulnerability by industry and vulnerability category.

SIDEBAR: GOVERNMENT AND FINANCIAL SERVICES: LOWER OVERALL VULNERABILITIES, HIGHER SERIOUS VULNERABILITIES

While vulnerability rates are remarkably similar across industries, Contrast Labs data indicates that two verticals have more than the others—government and financial services. In the government sector, 93% of applications have at least one vulnerability, and more than one-third (36%) have at least one serious vulnerability (Figures 8 and 10). The 93% figure is actually tied for the second lowest percentage among industries we surveyed, but the 36% figure is the highest in our list.

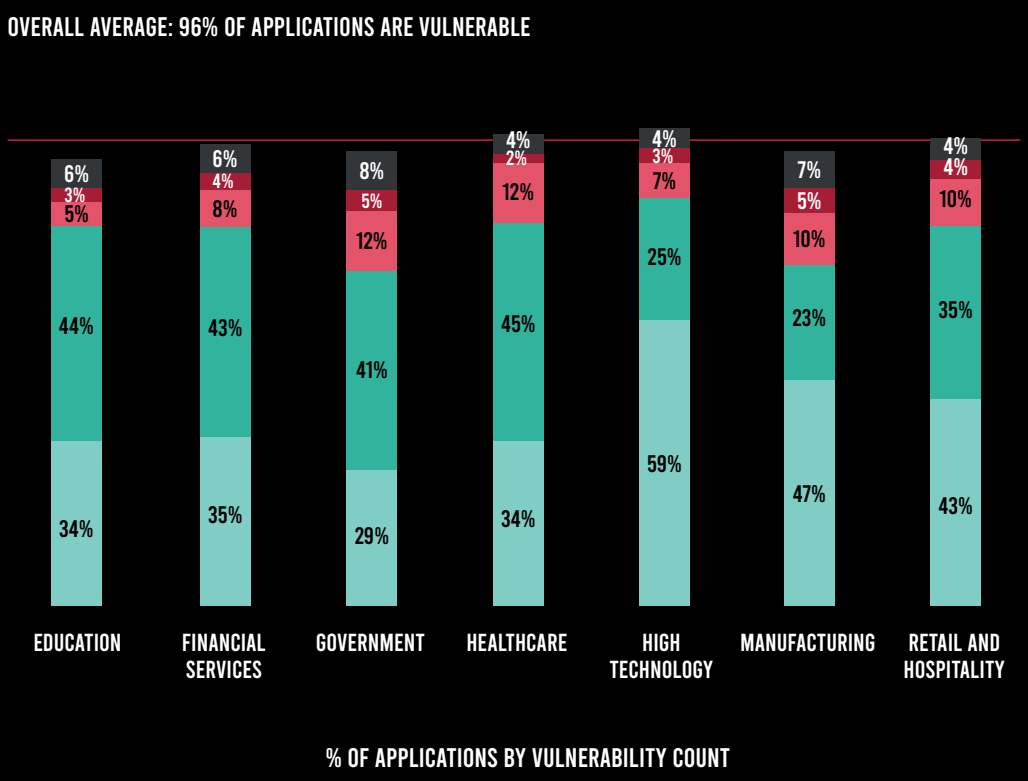


Figure 10. Percent of applications by vulnerability count, by application language and overall.

As a group, federal, state, and local government entities tend to use older technology than most industries in the private sector, and many use older languages in application development. This has been exemplified by the sudden shortage of COBOL programmers during the surge in unemployment claims caused by the COVID-19 pandemic.[3] Government entities are more likely to outsource development and therefore often

have difficulty preventing vulnerabilities through the contracting process. Also, agencies often cannot attract experts away from high-paying corporate security teams, making it difficult to hire an expert staff.

Financial services organizations also see serious vulnerabilities in more than 3 in 10 applications (32%), but have a middling percentage of applications with any vulnerability (95%). At first glance, this vertical appears to be the opposite of the government sector, deploying cutting-edge technology to win the business of increasingly tech-savvy consumers and businesses. Ironically, this level of innovation may work to these companies' detriment when it comes to software vulnerabilities, as they tend to try things that have not been tried before.[4] They also are burdened by sheer numbers—huge application and application programming interface (API) portfolios that change and adapt quickly. Growth by acquisition, platform migration, moves to the cloud and containers, and other digital transformation initiatives can make AppSec difficult for this industry.

06

REMEDIATION TIMELINES: ADDRESSING PROBLEMS MORE QUICKLY

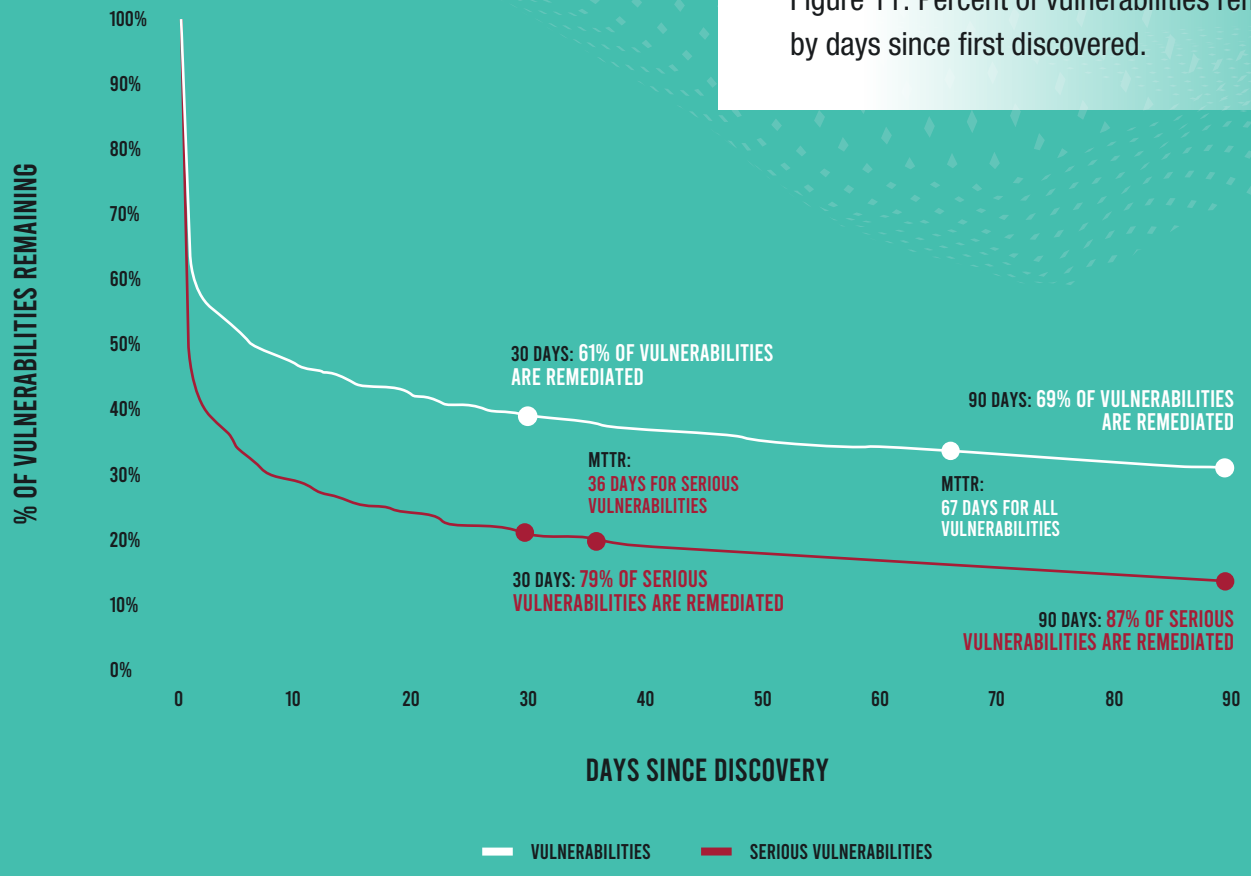
06 | REMEDIATION TIMELINES: ADDRESSING PROBLEMS MORE QUICKLY

Remediation timelines are a critical metric because the sooner a vulnerability is remediated, the less expensive and time-consuming the fix is. One study finds that the cost of remediating a vulnerability in an application already in production is 100 times the cost of addressing it during the design phase.[5] In addition, remediation that occurs late in the SDLC is likely to delay the rollout of the application.

Contrast Labs' data shows a mean (average) time to remediate (MTTR) of 67 days for all vulnerabilities and 36 days for serious ones (Figure 11). But those numbers tell only a small part of the story. The *median* time to remediate—the time at which 50% of vulnerabilities have been resolved—is just seven days, and 45% of all vulnerabilities and 62% of serious ones are resolved within just three days. Those numbers increase to 65% and 83% after 50 days.

One interesting note: Vulnerabilities that are not remediated within 30 days tend to remain after 90 days. Fully 79% of vulnerabilities and 65% of serious vulnerabilities not remediated within 30 days remain after day 90. In these instances, security debt accumulates over time if vulnerabilities are not remediated earlier in the SDLC—namely, the longer a vulnerability persists, the greater the likelihood that it will not be remediated. Unless there is an organizational culture or security-level agreement (SLA) policy that dictates a maximum age for vulnerabilities, this pattern will likely continue.

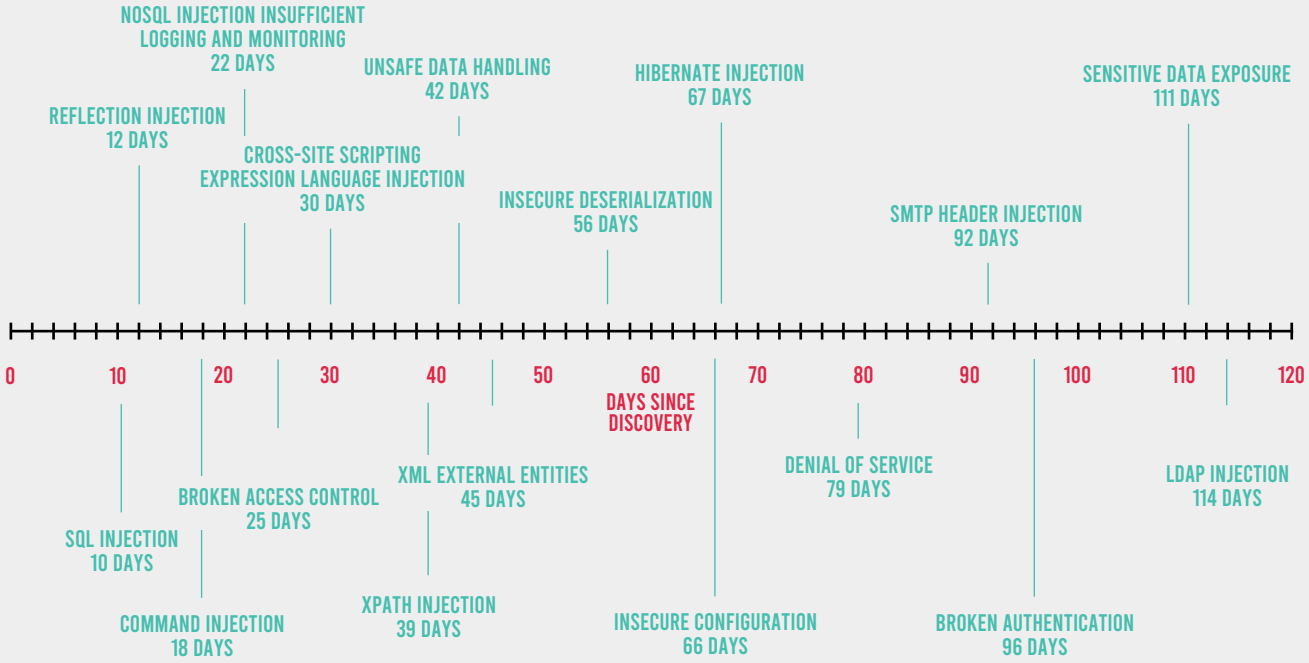
Figure 11. Percent of vulnerabilities remaining by days since first discovered.



BY VULNERABILITY TYPE

Going beyond the overall numbers, it becomes clear that the MTTR varies widely according to the type of vulnerability (Figure 12). SQL injection vulnerabilities, for example, are resolved in an average of 10 days, while vulnerabilities in the sensitive data exposure and broken authentication categories take more than three months to close. These differences can be at least partially explained by organizations' success in prioritizing vulnerabilities by risk. Contrast Security customers' MTTR for serious vulnerabilities is just over half the MTTR for all vulnerabilities.

Figure 12. Mean time to remediate (MTTR) vulnerabilities by vulnerability category.



BY INDUSTRY

When remediation data is analyzed by industry, interesting differences emerge (Figures 13 and 14). High technology (86%) and manufacturing (85%) companies resolve far more vulnerabilities in the first 30 days than education (43%) and government (50%). And these differences persist through at least day 100. Our takeaway: Different industries—and different individual organizations—are further along in the maturity of their DevSecOps integration than others, and these distinctions come out in time to resolution.

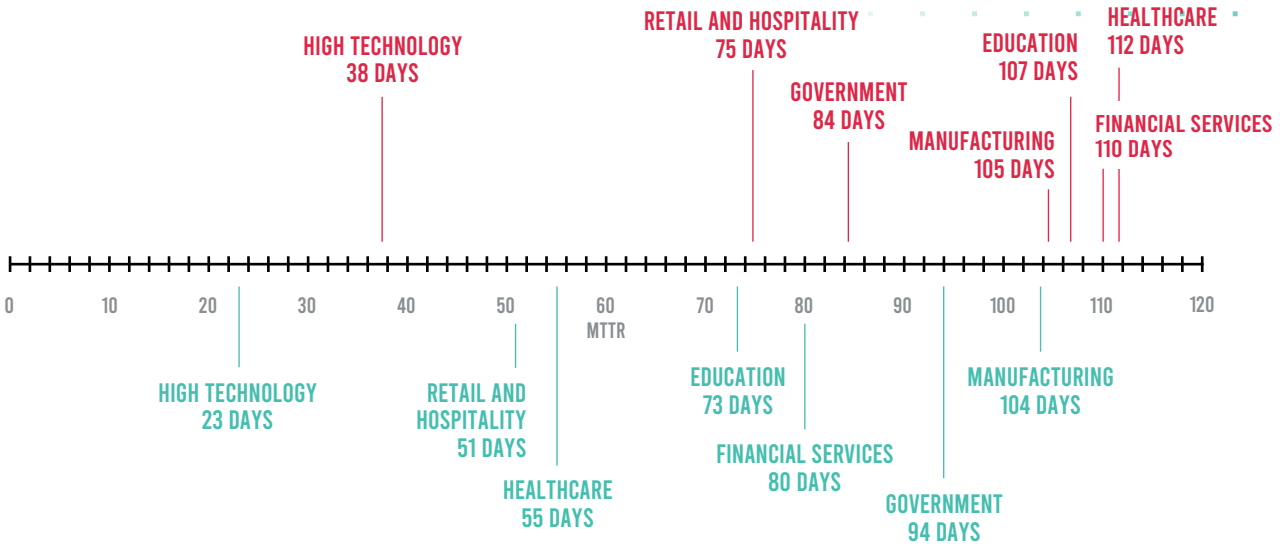


Figure 13. Mean time to remediate (MTTR) vulnerabilities and serious vulnerabilities by industry

Figure 14. Percent of all vulnerabilities and serious vulnerabilities remediated within 30, 60, and 90 days of discovery, by industry.

BY LEVEL OF SECURITY DEBT

Another lens through which to view time to remediation is the amount of security debt a company holds. Security debt is represented by the backlog of vulnerabilities that still need to be remediated across the organization.

These numbers vary widely: The mean backlog across all customers is 1,830 vulnerabilities as of June 1, 2019. The average organization sees 183 new vulnerabilities per month, three per application. However, 83% of applications see two or fewer vulnerabilities introduced per month.

Notwithstanding, the mean backlog does not reflect the fact that a relatively small number of applications represent a relatively large percentage of the backlog. When considering the subset of customers whose backlog is below the average of 1,830 vulnerabilities, the mean backlog is just 428 vulnerabilities—or 17 per application. And organizations with below-average security debt see fewer new vulnerabilities as well—just 68 per month on average rather than 183 (translating to 1.7x lower risk).

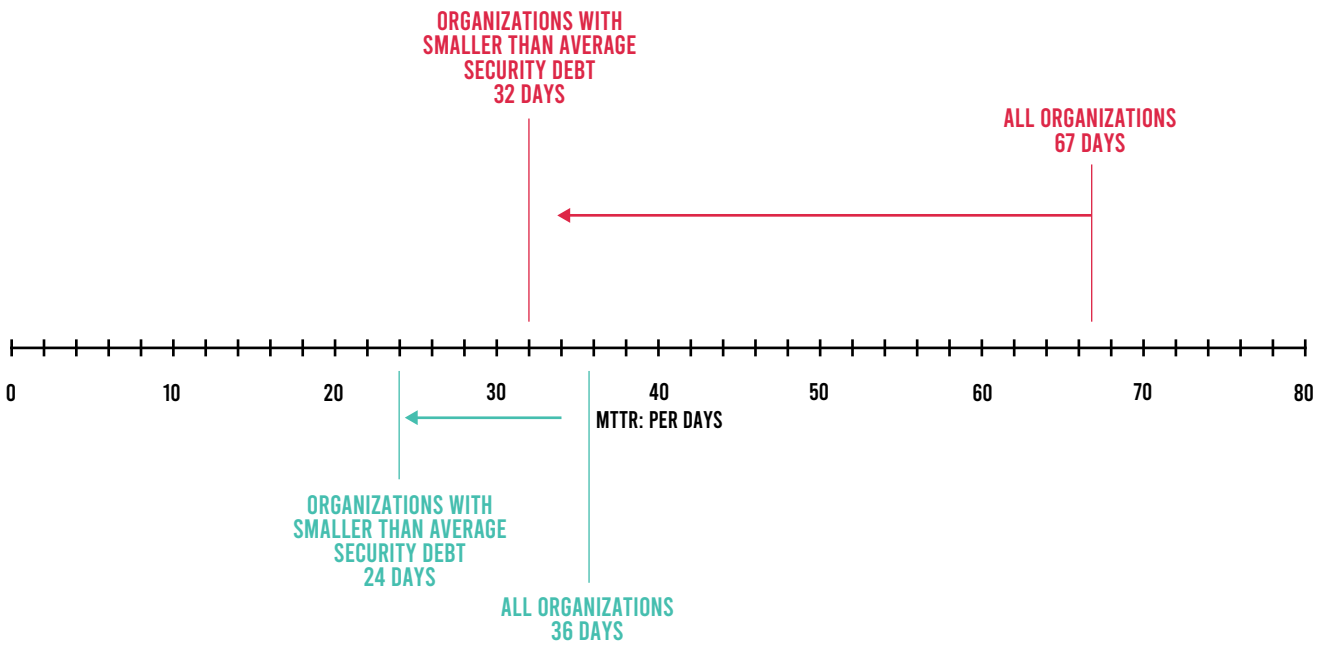


Figure 15. MTTR vulnerabilities and serious vulnerabilities for all organizations and organizations with smaller than average security debt.

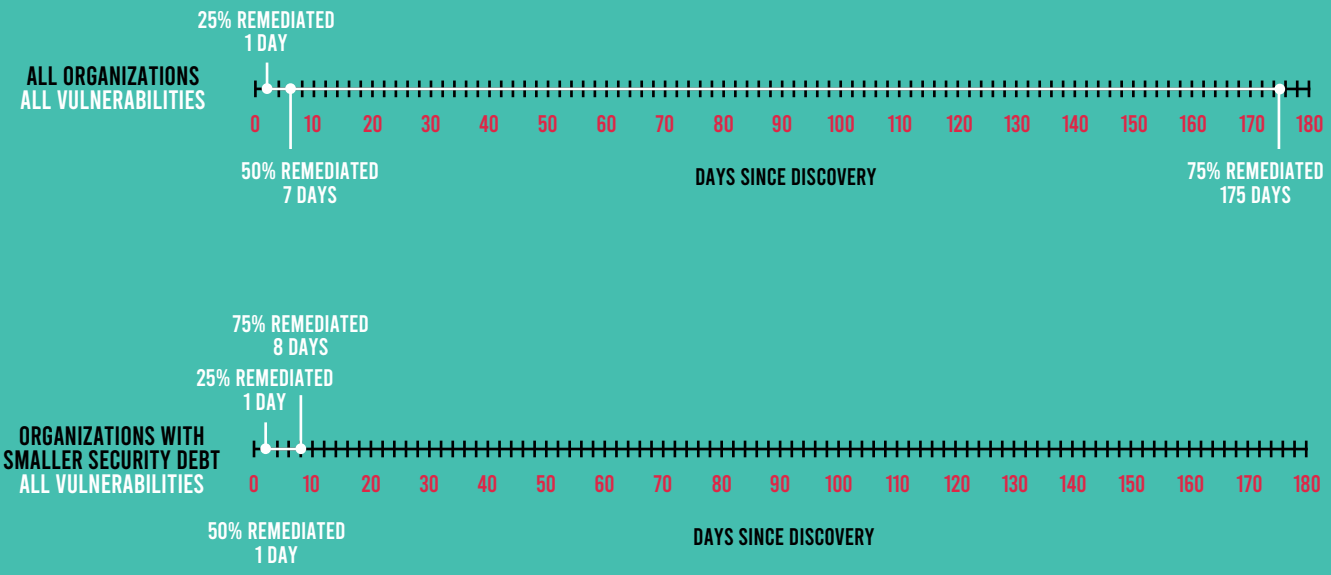


Figure 16. Days since discovery for all vulnerabilities remediated by all organizations versus those with smaller than average security debt.

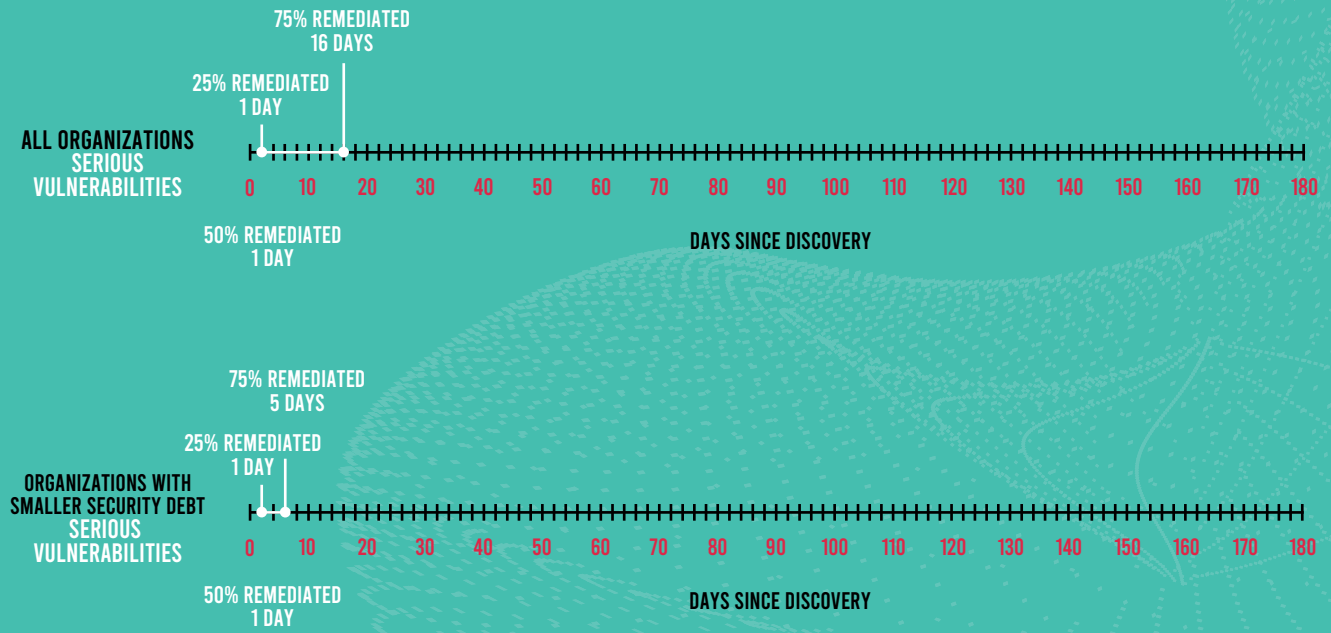


Figure 17. Days since discovery for serious vulnerabilities remediated by all organizations versus those with smaller than average security debt.

The differences extend to MTTR. Organizations with below-average security debt have an MTTR of 32 days—less than half the MTTR for all organizations (Figure 15). And the time it takes to remediate 75% of serious vulnerabilities is five days for those with below-average security debt, compared with 16 days for all organizations (Figure 17).

Companies with high security debt face operational risk from cyber exploits and legal exposure from potential release of customer information.[6] When an organization faces such a challenge, one solution is to implement a runtime application self-protection (RASP) solution such as Contrast Protect, which can provide highly effective stopgap protection against exploitation of unresolved vulnerabilities in running applications.[7]

SIDEBAR: REMEDIATION SPEED IN CONTEXT

Three things should be noted with regard to the remediation data from the Contrast customer community. First, the remediation speed calculated for June 2019 through May 2020 is much improved over the 2018 data from Contrast Labs.[8] This suggests that remediation rates are improving as the Contrast Security customer base grows, and as organizations fully integrate its solutions into their SDLC.

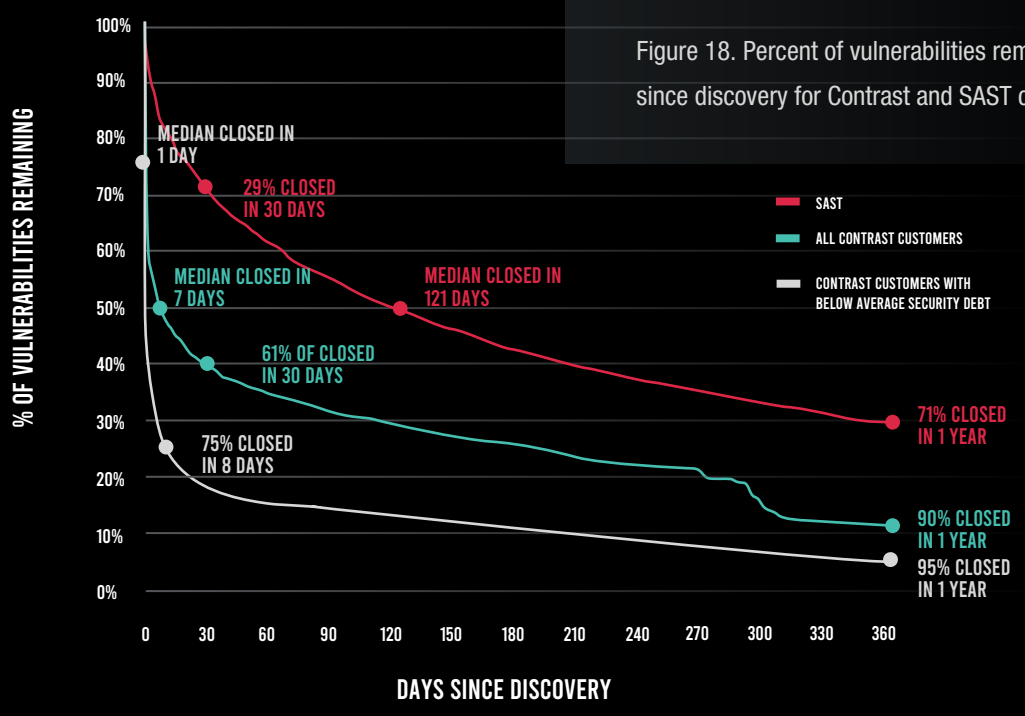


Figure 18. Percent of vulnerabilities remaining since discovery for Contrast and SAST customers.

Second, Contrast customers' remediation performance continues to outpace other industry players. As an example, the latest report from one static application security testing (SAST) vendor[9] shows an MTTR for all application vulnerabilities of 171 days—compared with 67 days for Contrast Security customers. And they achieve a median time to remediate of seven days as compared with 121 days for SAST (Figure 18).[10]

The differences remain significant when compared over a period of 90 days (Figure 19). And after a full year, only 10.3% of vulnerabilities (and only 2.3% of serious vulnerabilities) remain unresolved for Contrast customers. The SAST customer base, on the other hand, sees 29.4% of vulnerabilities still not remediated after a full year. The differences are even more dramatic when looking at serious vulnerabilities. Contrast customers see 25% of serious vulnerabilities remediated in one day and 75% in 16 days, as compared with 19 days and 292 days, respectively, for SAST (Figure 20).

Third, for Contrast customers with below-average security debt (Figure 17), closed median for vulnerabilities remediated is one day (Figure 18). And 75% of vulnerabilities are remediated within eight days. These numbers are dramatic and confirmation that organizations with reduced security debt have significantly lower risk. A comparison of Contrast customers with below-average security debt versus organizations using SAST tools shows a huge difference—121x faster for median remediation time and 45x faster for remediation of 75% of vulnerabilities.

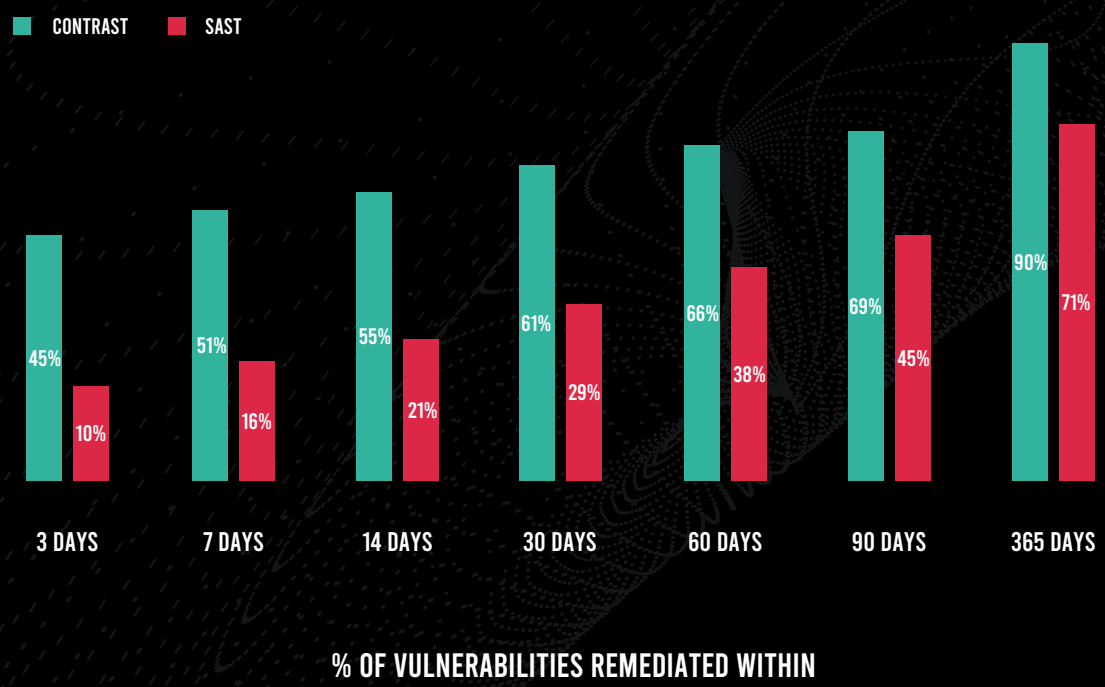


Figure 19. Percent of vulnerabilities remediated within 3, 7, 14, 30, 60, 90, and 365 days using Contrast and SAST.

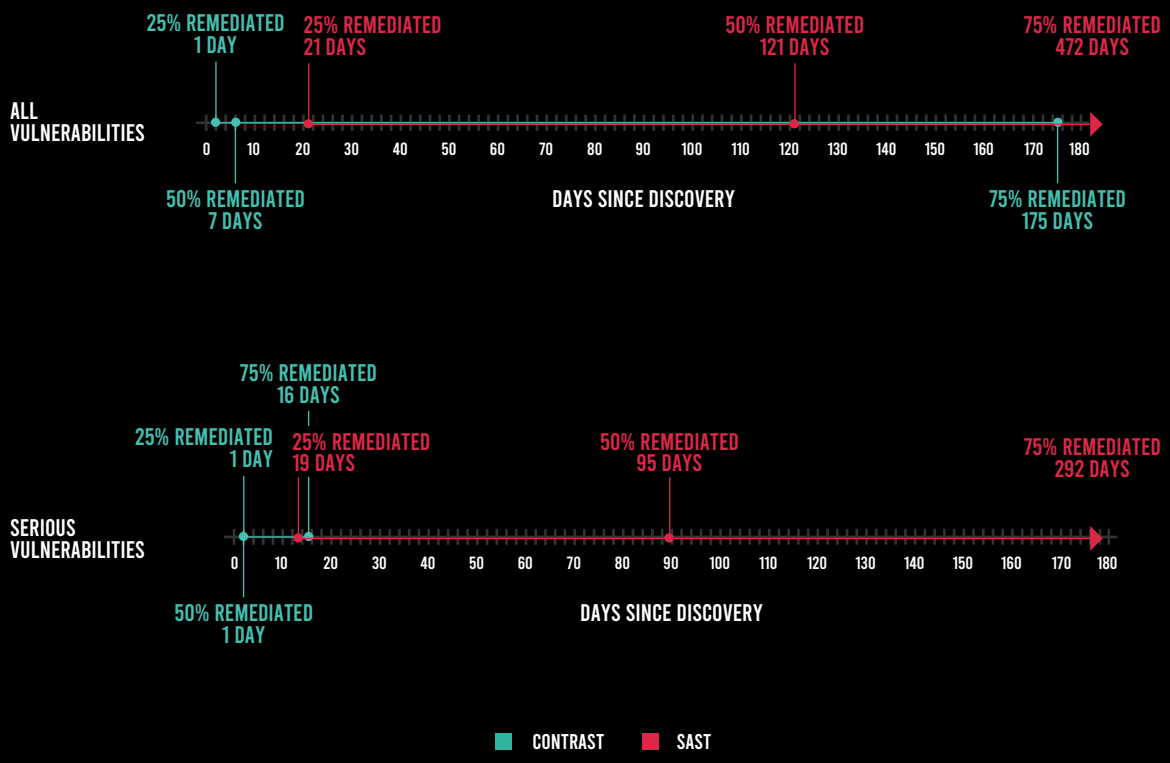
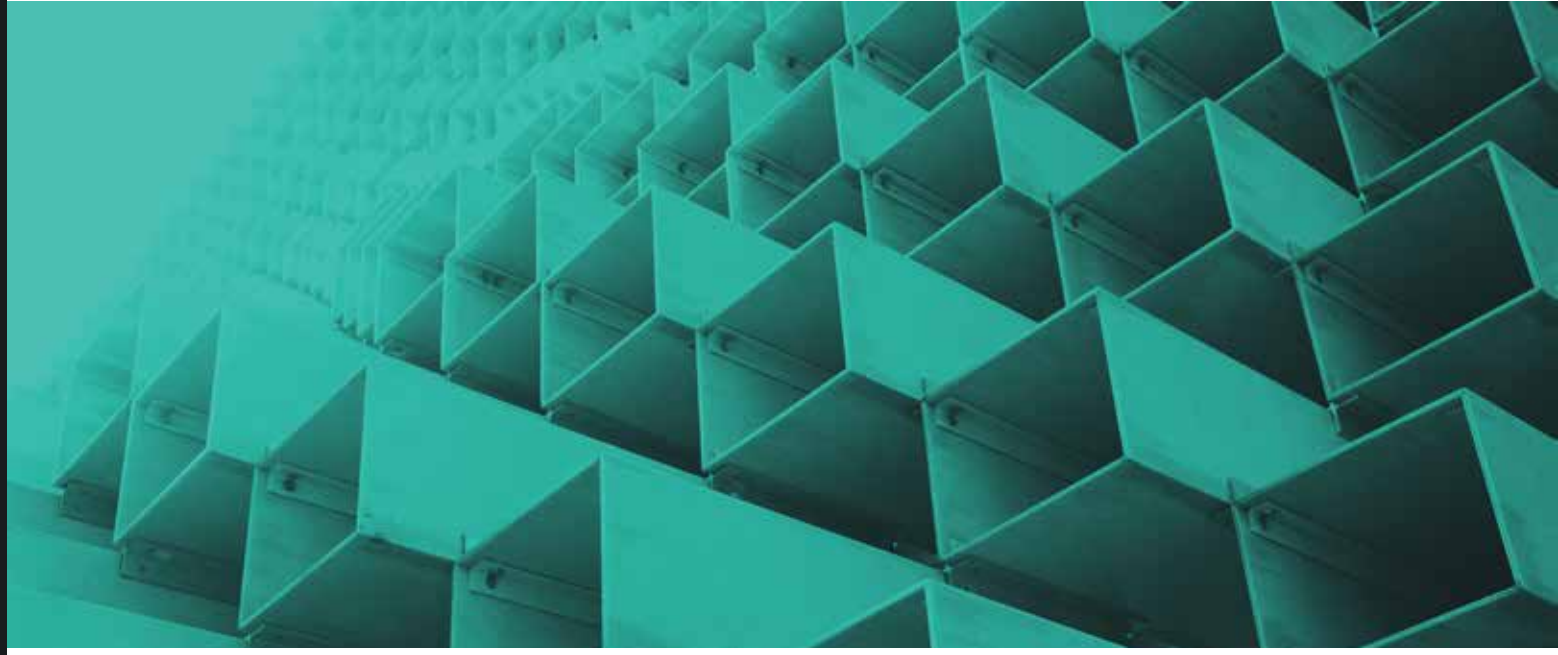


Figure 20. Days since discovery when 25%, 50%, and 75% of vulnerabilities and serious vulnerabilities are remediated using Contrast versus SAST.

07

OPEN-SOURCE LIBRARIES: DRIVING INNOVATION AND INTRODUCING RISK



07 | OPEN-SOURCE LIBRARIES: DRIVING INNOVATION AND INTRODUCING RISK

The development community is driving further efficiencies through an increased use of open-source code. In fact, Forrester recently found a 40% jump in the use of open-source code in one year.[11] At the same time, the number of vulnerabilities logged in the CVE database is skyrocketing at an unprecedented clip.[12]

Contrast Labs telemetry verifies the prevalence of open-source code in organizations that use Contrast OSS. The average application contains code from 32 different libraries, and 14 of those libraries on average are active—that is, invoked by the application (Figure 21). The latter figure may be a surprise to some readers, as software composition analysis (SCA) tools typically only report on the presence of a library in an application. But this data from Contrast OSS shows that 56% of libraries present in an average application are never invoked and create almost no risk. It is safe to ignore the code from these libraries until a time when it is reasonable to do an update. Instead, organizations can prioritize vulnerabilities in code that is actually run by the application.

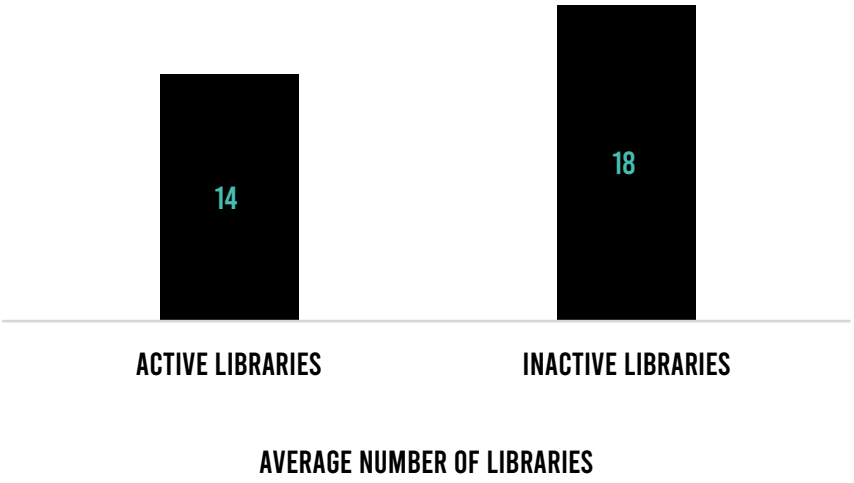


Figure 21. Average number of libraries per application.

With such a large number of libraries in use and the burgeoning volume of CVEs, version control is important. Some companies try to ensure the latest “stable” version of each library is used in each application. This is because successive versions of major libraries are released with remediation for newly discovered CVEs, and a large percentage of potential vulnerabilities can therefore be avoided by simply using a later version of each library.

While there are often good reasons not to use the very latest version of a library (see sidebar: "Challenges for Open-source Version Control," page 34), developers tend to be many versions behind with the code they use—especially in the Java programming language. Overall, among all instances of library use, only 14% used the latest version of the library in question, but that figure is 17% for .NET applications and only 9% for Java applications (Figure 22). Worse yet, when the top 10 Java libraries are used in applications covered by Contrast OSS, nine of them use a version that is at least a dozen versions behind the latest revision (Figure 23). Slightly better news can be found with .NET, where many of the top libraries are just a handful of versions behind the latest.

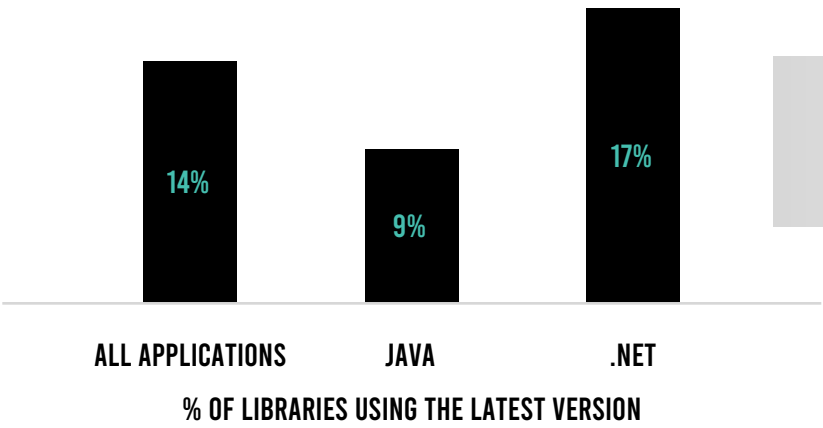
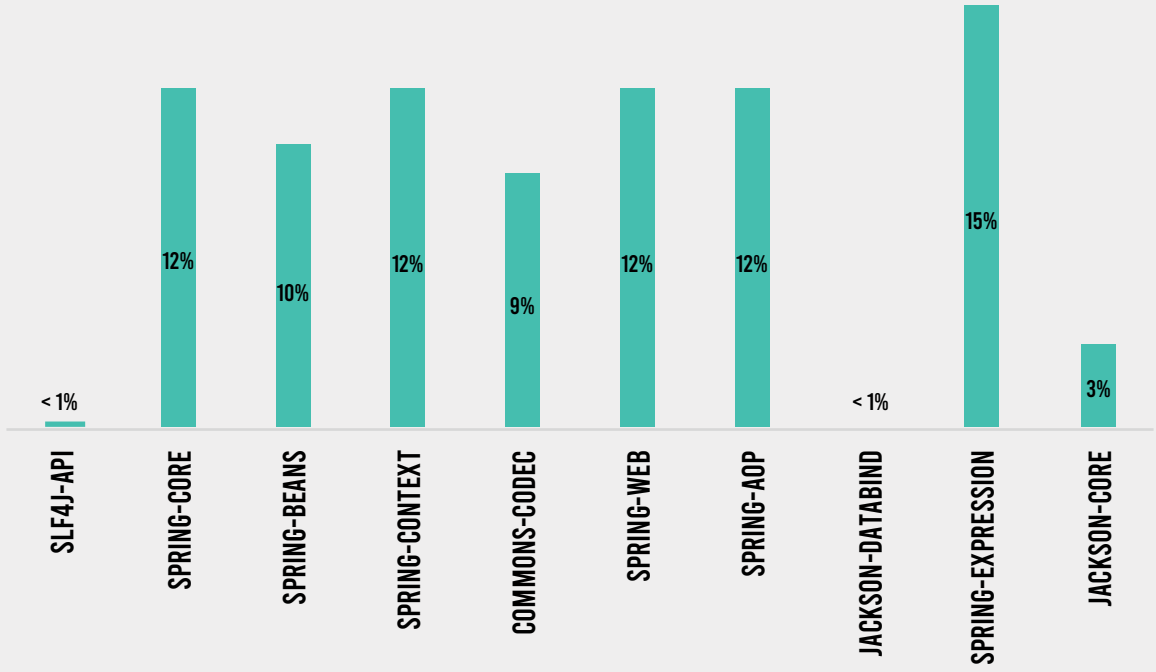
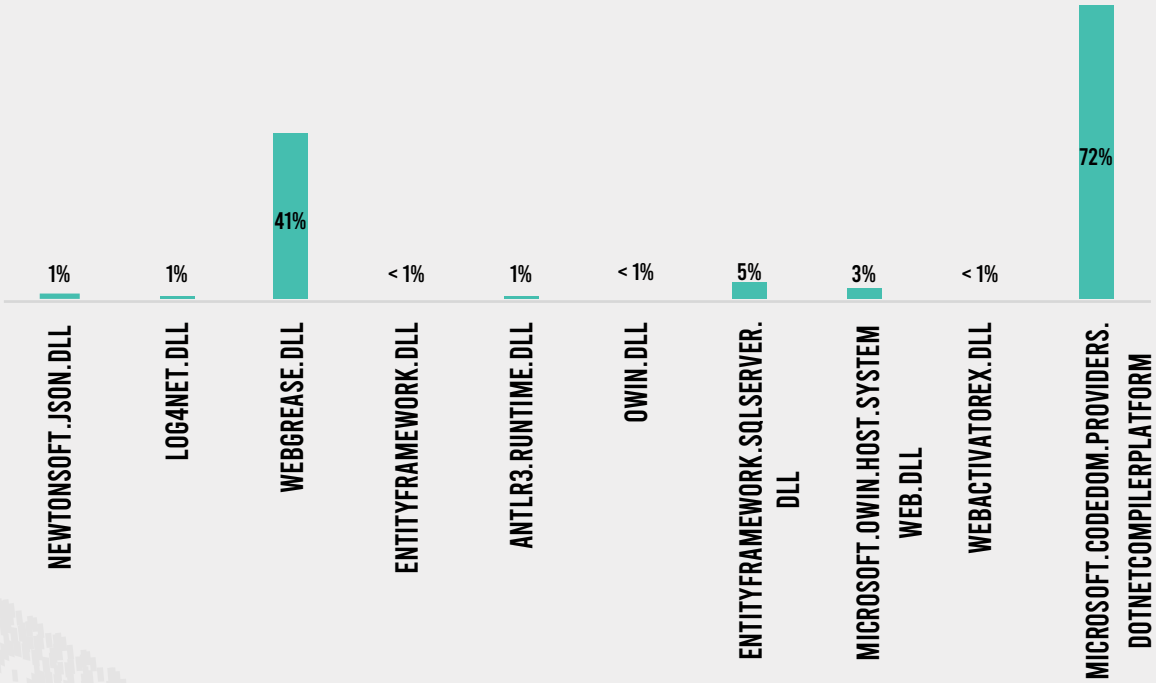


Figure 22. Average number of libraries using the latest version.



% OF APPLICATIONS USING THE LATEST VERSION OF THE TOP 10 JAVA LIBRARIES



% OF APPLICATIONS USING THE LATEST VERSION OF THE TOP 10 .NET LIBRARIES

Figure 23. Percent of applications using the latest version of the top 10 Java and .NET applications.

The average application across all languages contains four CVEs—just over two that are rated either High or Critical, which is concerning. All of the 10 most commonly found CVEs for Java were identified during 2019, but only one of the top 10 .NET vulnerabilities is that recent, and others go back as far as 2014. Java CVEs are more serious as well, with nine of the top 10 receiving a CVSS score of 9.8 or higher. The top 10 .NET vulnerabilities, on the other hand, all have a CVSS score below 7.6.

SIDEBAR: CHALLENGES FOR OPEN-SOURCE VERSION CONTROL

In general, the best strategy for developers is to keep open-source libraries as up to date as possible. However, updating libraries can bring risk to an organization as well. While using the latest version of a library will address the vast majority of vulnerabilities that would be in earlier versions of the code, doing so has the potential of breaking something in the application. Having something stop working in an application after something else is updated is not only frustrating to developers but can introduce business risk for the organization. Thus, the decision on whether to update a library must take into account not only AppSec considerations but the organization's overall risk management portfolio.

Organizations have four acceptable choices when it comes to version decisions for open-source code:

1. Choose the earliest available version of a library that addresses a specific vulnerability fixed, regardless of how old it is. The upside is that minimal changes are required to the software. The downside is that the process may have to be redone if another vulnerability is found.

2. Choose the latest "stable" version of a library. The problem is that this is not well-defined. Some libraries pre-release versions that are not yet ready for prime time.

3. Something in between options 1 and 2.

4. Do nothing, and use a RASP solution to prevent vulnerabilities from being exploited. This option works for specific CVEs covered by the product.

SIDEBAR: WHY SO MANY NEW NEWER CVES?

Since the CVE database was introduced in 1999, no more than 8,000 new vulnerabilities were added to it in a single year until 2017, when nearly 15,000 were added. More than 16,000 were added in 2018 and more than 12,000 in 2019.[13] Why the sudden surge? Several factors probably contribute to the phenomenon:

- An increase in the number of libraries covered by the database,[14] corresponding with a sharp increase in the use of open-source code by developers[15]
- A simplified application process that makes it easier for developers to report vulnerabilities for inclusion in the database[16]
- The increasing popularity of bug-bounty programs, which account for 8% of reported vulnerabilities[17] and saw a 27% increase in average payouts per bounty paid last year[18]

As the number of published vulnerabilities in open-source libraries proliferates, it becomes increasingly important that developers use recent, stable, and secure versions of every library, ensuring that as many CVEs as possible are remediated before the code hits an application.

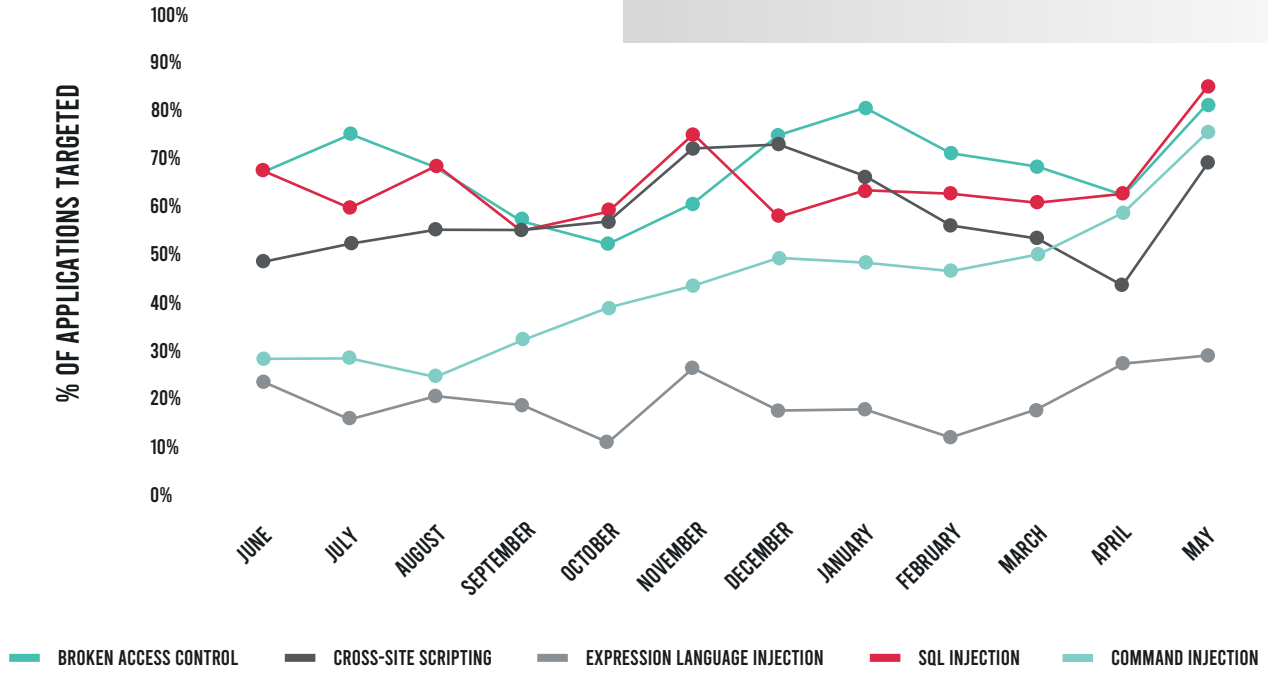
08

ATTACKS: RELENTLESS VOLUME

08 | ATTACKS: RELENTLESS VOLUME

A well-known characteristic of today's overall threat landscape is an ever-increasing volume of attacks, and application- and API-layer attacks are no exception. Contrast Labs finds that applications in production were pummeled throughout 2019 with an average of 13,279 attacks per application per month. An astounding 65% of applications were targeted by SQL injection attacks, 62% by broken access control attacks, and 54% by XSS attacks.

Figure 24. Percent of applications targeted by attacks on specific vulnerabilities each month.



For the five most common attack types, the volume of attacks has trended upward over the past 12 months—all near their 12-month high in May 2020 (Figure 24), with an average increase of 60% over the 12-month period. The percentage of applications targeted with command injection attacks increased by 79% over the past year—from 28% to 78%. Nearly 8 in 10 Java applications and 5 in 10 .NET applications endured SQL injection attacks, while well over half of applications in both languages saw broken access control and XSS attacks (Figure 25).

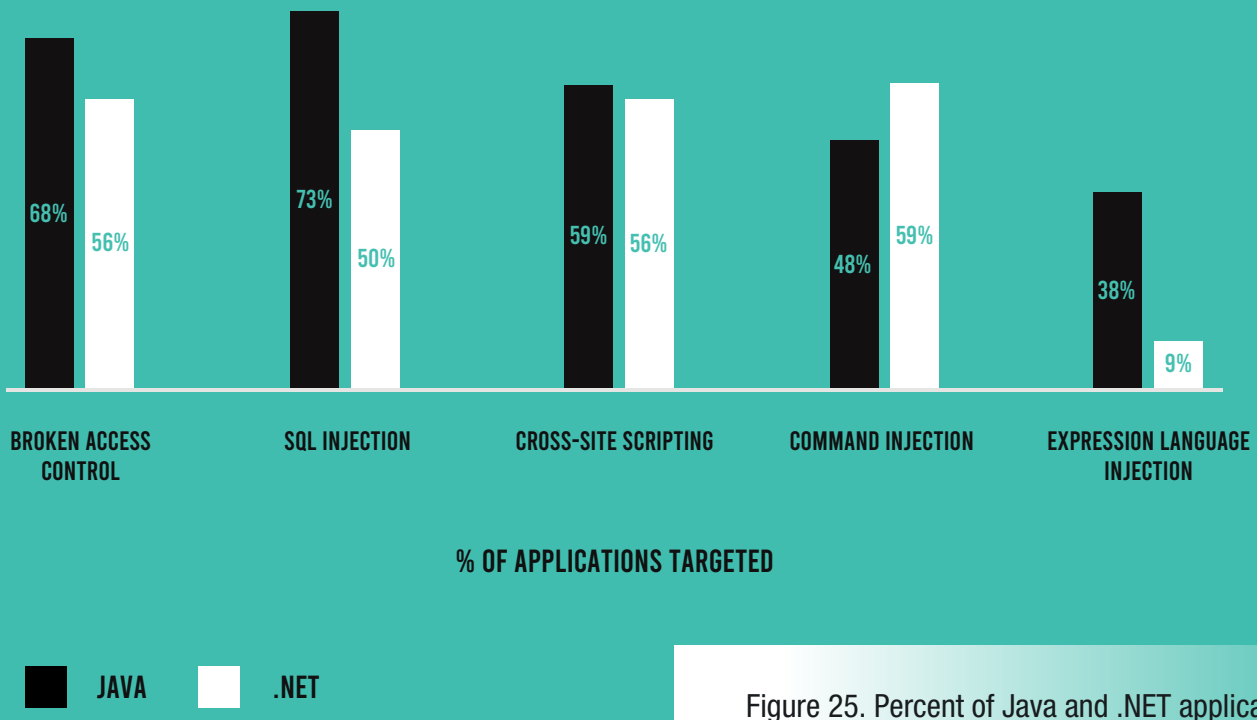


Figure 25. Percent of Java and .NET applications targeted by attacks on specific vulnerabilities.

Not surprisingly, these three most common attack types match the three most common serious vulnerabilities identified during development—albeit in a different order of prevalence (Figure 26). Applications released into production with vulnerabilities in these categories are exposed to the high likelihood of a successful attack.

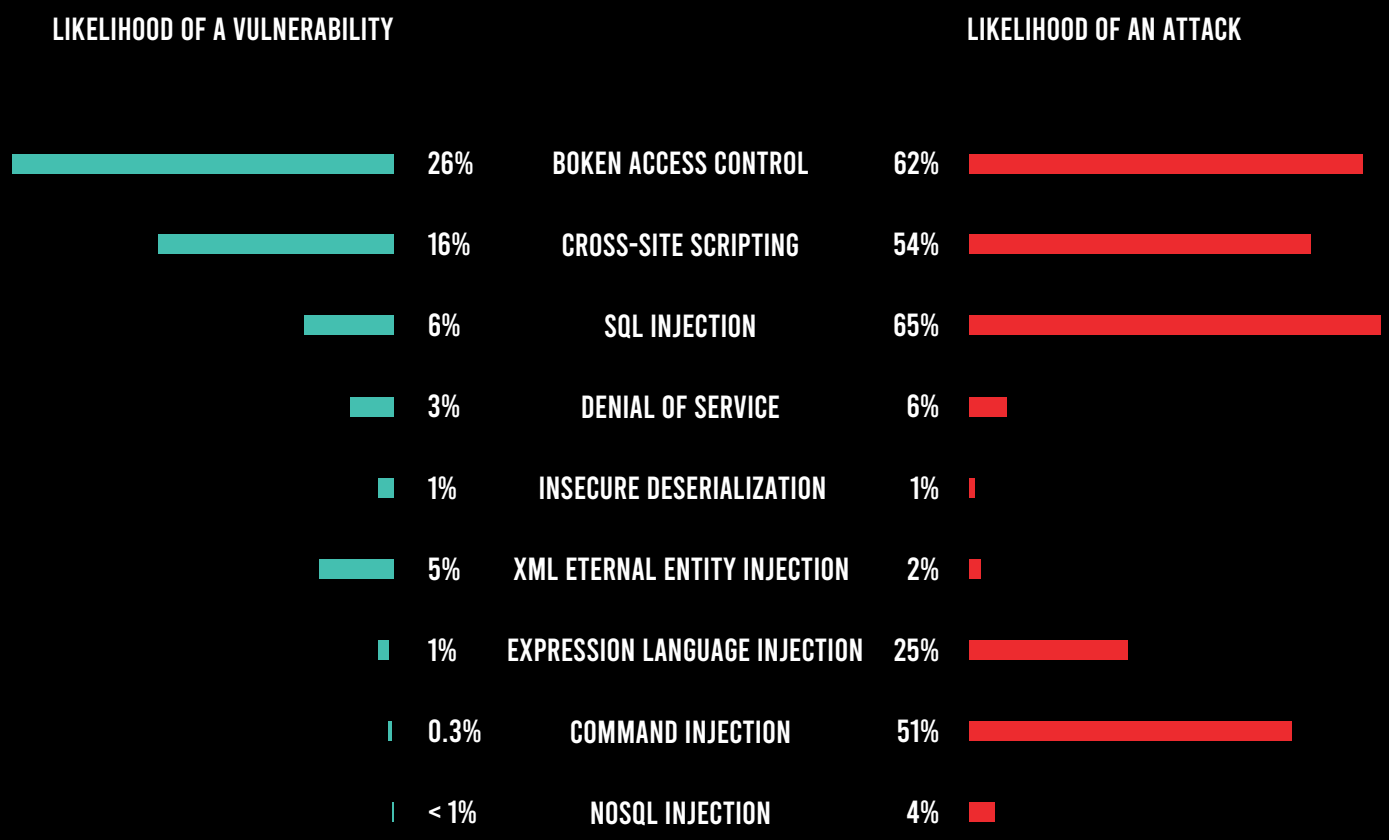


Figure 26. Likelihood of vulnerability and attack types.

BY EXPLOITABILITY

Fortunately, 98% of these attacks target vulnerabilities that are not present in a given application, but this percentage is not consistent across all attack types. For example, command injection and SQL injection attacks targeted very few exploitable vulnerabilities—0.3% and 1%, respectively. This results in less risk associated with the attack types even though they make up the highest percentage of attacks and the first and fourth most common attack types.

BY INDUSTRY

Threat actors execute an extremely high volume of attacks because they realize that only a small percentage of them will be successful. Yet, they also target more attacks on the most common serious vulnerabilities, which in effect increases the odds that a small subset of attacks will be successful. As a result, all industries see the highest volume of attacks in broken access control, command injection, XSS, SQL injection, and expression language injection (Figure 27).

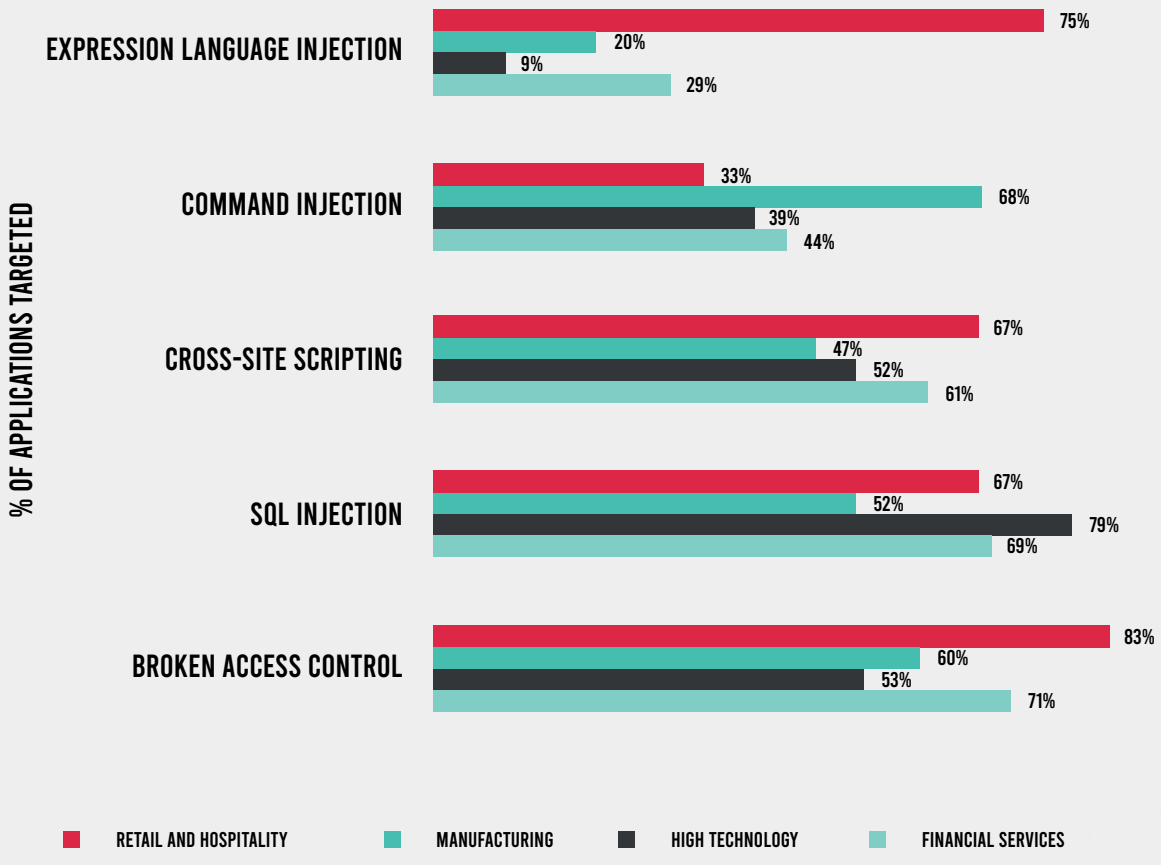


Figure 27. Percent of applications targeted by attacks on specific vulnerabilities by industry.

As to specific industry attack trends, retail and hospitality organizations were unusually targeted by broken access control attacks, which impact 83% of applications in that vertical. Expression language injection attacks target 75% of retail and hospitality applications—46 percentage points higher than any other industry. Fortunately, this is a rare vulnerability in this industry, occurring in just 1% of applications.

SQL injection attacks are most prevalent at high technology and financial services organizations, and manufacturing organizations see a high proportion of command injection attacks.

SIDEBAR: INSECURE DESERIALIZATION

Serialization is the process of converting an object into a format that can be transmitted via file system or network. The reverse process is called deserialization—rebuilding serialized data back into the original objects so they can be used by web applications. This is where the risk comes in. If the right controls are not in place, attackers can serialize malicious objections that look safe until they are deserialized, resulting in a denial-of-service (DoS) attack or other intrusion. Insecure deserialization is the seventh most prevalent serious vulnerability for Contrast customers, and only 1% of applications reported attacks.

While not as common as it used to be, this can still be dangerous. Organizations should especially pay attention to Java applications, where many deserialization risks continue to persist. And security and operations teams can always enable RASP to sandbox deserialization to prevent harmful things from happening during object hydration (viz., infiltration of data with malicious code embedded).

SIDEBAR: PUTTING IT ALL TOGETHER: THE TOP SIX APPLICATION RISKS BASED ON REAL-WORLD DATA

Analysis of data from actual Contrast customers provides Contrast Labs with a unique perspective on the threats faced by development and security teams today. Based on the risk they present to organizations, Contrast Labs identified the top six AppSec risks based on its understanding of custom-code vulnerabilities, open-source vulnerabilities, and real attack data from production. Based on that information, we would like to propose some strategic recommendations:

- SQL Injection. A classic AppSec issue, SQL injection should still be considered as one of the top risks. SQL injection attacks were among the three most prolific attack types for the entire year. The prevalence, severity, and availability of free and easy testing tools of SQL injection have attracted attackers for decades. Contrast Security customers take SQL injection very seriously as they prioritize vulnerabilities to remediate, and as a result, this vulnerability category has the shortest MTTR.

  While Contrast Labs finds SQL injections' exploitability rate to be just 1%, the high attack volume means that tens of thousands of SQL injection attempts are successful every year. SQL injection vulnerabilities are frequently rated as serious, partly because of the impact they can have when they are successful. When a cyber criminal infiltrates a database, the result can be devastating for an organization. To reduce risk, organizations should ban the use of nonparameterized queries and strongly consider adding RASP to protect applications in production.

- Broken Access Control. Authorization is one of the most complex vulnerabilities. Because most authorization schemes are custom to a particular application, it is very difficult for automated tools and even human penetration testers and code reviewers to detect these vulnerabilities. It is similarly hard to detect these attacks in production. As a result, broken access control is still near the top of our vulnerability prevalence and attacker focus lists. Accordingly, Contrast Labs finds that broken access control is the second most prevalent serious vulnerability type and the second most common vulnerability targeted by an attack.

  Broken access control is a tremendously easy vulnerability for developers to introduce, as getting it right requires a lot of discipline across the entire codebase. The consequences of a successful attack can range from minor data leaks to full access to administrative functions. Contrast recommends centralizing access control code mechanisms, using standard framework mechanisms when possible,

creating simple idioms for developers to follow, and deploying continuous testing to assure that access controls are in place and effective.

- Cross-site Scripting (XSS). This is another classic AppSec vulnerability that has remained in the OWASP Top 10 since it was instituted. It is the most prevalent serious vulnerability in Contrast customers' applications, and the third most common vulnerability targeted by an attack. But while the vulnerability rates and attack rates are quite high, the numbers show that there are a small number of applications with very high XSS occurrence rates driving the numbers up.[19]

  When an application has one XSS vulnerability, XSS is most likely very prevalent throughout that application. The impact of XSS is not as serious as the other vulnerabilities listed here, as a successful XSS attack results only in compromising a single user's browser. Still, a crafty attacker might be able to worm an XSS or perform a mass attack, as with the Samy worm.[20] To avoid this vulnerability, developers should be trained to use frameworks properly and encode HTML output correctly.

- Expression Language (EL) Injection. EL injection affects OGNL and other expression languages, and is a newer vulnerability type compared to others on the list. It is very attractive to attackers because a successful exploit leads to a complete host takeover. EL injection gained a lot of notoriety for causing several extremely large and public breaches, and they are rated as high severity vulnerabilities by Contrast Labs.

  Only 1% of applications in Contrast Labs' dataset had EL injection vulnerabilities over the past year, but 25% of applications recorded attacks targeting the vulnerability. In addition, several known EL injection CVEs in open-source frameworks such as Struts and Spring are heavily targeted by attackers. There are also almost certainly a number of these vulnerabilities that have not been discovered. Nothing can be done about EL injection except to keep frameworks up to date and implement RASP protection to sandbox EL evaluation to prevent potential malicious attacks.

- Command Injection. Command injection attacks are the second most common attack type, but the exploitability for command injection is just 0.3%. Even though it is obviously a favorite of attackers, there are exceedingly few vulnerabilities out there to find. Despite all the effort, command injection attacks were extremely unlikely to target vulnerable code in the target applications.

Despite the extremely low odds of success, adversaries keep trying because of the potential payoff. A successful attack could result in a complete takeover of the entire host and all the code and data on it. Command injection attacks are the most direct route to a massive compromise: If successful, they provide a hacker with the ability to run commands against the underlying system and enable them to possibly establish a foothold in the infrastructure to pivot horizontally to other systems as well. As a potential prevention, organizations should consider banning the direct use of OS commands and implementing RASP to sandbox their execution.

- XML External Entity (XXE) Injection. XXE injection is the fourth most prevalent serious vulnerability, according to Contrast Labs data. Many applications and particularly APIs parse untrusted XML documents and unless they take specific actions, attackers can access internal files, URLs, and possibly even execute commands on the host system. XXE injection is somewhat overlooked by attackers today, with just 2% of applications reporting an attack. But as APIs are increasingly targeted by attackers, Contrast Labs expects to see the number of public breaches based on XXE injection increase.[21]

  To offset the risks, organizations should consider a policy disallowing the use of XML parsers unless doctype processing is disabled. However, there is nothing an organization can do about this vulnerability in open-source libraries and frameworks except to keep them up to date and to enable RASP to prevent XML documents with external entities from being parsed.

09

CONCLUSION

09 | CONCLUSION

The “2020 Application Security Observability Report” contains real-world data from real organizations. Based on the aforementioned data insights, Contrast Labs offers the following takeaways and recommendations:

DEALING WITH VULNERABILITIES:

- *Vulnerabilities should continue to be a key focus for security and development teams*, with more than one in 10 having six or more serious vulnerabilities.

- *Understanding the risk level of each vulnerability* helps security and development teams to prioritize the most important fixes based on the probability that it could cause problems. For example, the realization that sensitive data exposure vulnerabilities are high volume but relatively low in risk (Figure 2) can help organizations prioritize other, more risky vulnerability types.

- *Identifying and remediating vulnerabilities early in the process* helps organizations avoid the time and cost of remediating them later. Contrast Security customers resolve 79% of serious vulnerabilities in the first 30 days (Figure 11).

- *Addressing vulnerabilities before moving an application into production* is key, as the high volume of attacks means that any remaining vulnerabilities put an organization at great risk. The volume is trending upward for each of the five most common attack types (Figure 24). Command injection attacks more than tripled in prevalence over the past year.

MOVING BEYOND LEGACY APPROACHES TO APPSEC:

- *Practicing careful version control with open-source libraries* is critical, as it is best to use the library version that addresses the most CVEs without potentially causing problems in other areas. Organizations still have significant room for improvement in this regard (Figure 22).

- *Decreasing security debt* not only reduces stress on development, security, and operations teams but it also greatly improves time to resolution and actually results in fewer new vulnerabilities being identified. This can be accomplished with a combination of continuous testing, RASP protection for applications in production that still have vulnerabilities, and clearing backlogged vulnerabilities according to the risk they pose.

- *Balancing security efforts across the SDLC* improves greatly on the traditional model of conducting a big security test immediately before deployment. Instead, the best model shifts left by empowering developers to participate effectively through continuous testing, and extends right into production with visibility and exploit protection.

As the threat landscape becomes increasingly risky for organizations,[22] it is imperative that corporate software applications work properly and do not allow for intrusions of any kind. Companies that take a strategic, comprehensive approach to AppSec security do so because they have detailed data to help them succeed—data that is only provided by tools from Contrast. These organizations are setting themselves up for success—in delivering secure applications, and in protecting and enhancing the bottom line.

CONTRIBUTORS

JEFF WILLIAMS

CTO AND CO-FOUNDER,
CONTRAST SECURITY

Jeff brings more than 20 years of security leadership experience as Co-Founder and Chief Executive 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.

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 "OWASP Top Ten," Open Web Application Security Project, accessed May 18, 2020.
2 Christopher Mims, "Every Company Is Now a Tech Company," The Wall Street Journal, December 4, 2018.
3 Alicia Lee, "Wanted urgently: People who know a half century-old computer language so states can process unemployment claims," CNN, April 8, 2020.
4 Randy Bean, "How FinTech Initiatives Are Driving Financial Services Innovation," Forbes, July 10, 2018.
5 Mukesh Soni, "Defect Prevention: Reducing Costs and Enhancing Quality," iSixSigma, accessed April 9, 2020.
6 Taylor Armerding, "What is security debt, and how do I get out of it?" Security Boulevard, March 16, 2020.
7 "State-of-the-Art RASP Is AppSec Exactly Where It Is Needed—in Production Runtimes," Contrast Security, accessed June 16, 2020.
8 Katharine Watson, "How Do Teams Stay Afloat in an Ocean of Vulnerabilities? They Remediate Faster (3.0X Faster!)," Contrast Security Blog Post, May 6, 2019.
9 "State of Software Security X," Veracode, accessed May 18, 2020.
10 "State of Software Security Volume 9," Veracode, accessed May 18, 2020.
11 Amy DeMartine and Jennifer Adams, "Application Security Market Will Exceed $7 Billion By 2023," Forrester, updated March 29, 2019.
12 Liam Tung, "Open-source security: This is why bugs in open-source software have hit a record high," ZDNet, March 13, 2020.
13 "Security Vulnerabilities," CVE Details, accessed May 18, 2020.
14 Robert Lemos, "The state of vulnerability reports: What the CVE surge means," TechBeacon, accessed May 18, 2020.
15 Amy DeMartine and Jennifer Adams, "Application Security Market Will Exceed $7 Billion By 2023," Forrester, updated March 29, 2019.
16 "What is the reason for the increase in CVEs since 2017?" StackExchange, February 14, 2019.
17 Robert Lemos, "The state of vulnerability reports: What the CVE surge means," TechBeacon, accessed May 18, 2020.
18 Nate Swanner, "Bug Bounty Payouts Way Up as Companies Rush to Patch Holes," Dice, August 5, 2019.
19 David Lindner, "Contrast Labs: Mapping Risk Profiles for Select OWASP Top 10 Vulnerabilities to Understand Their AppSec Risk," Contrast Security Blog, May 19, 2020.
20 "Samy (computer worm)," Wikipedia, accessed June 16, 2020.
21 "The State of API 2019 Report," SmartBear, accessed June 16, 2020.
22 Patrick Spencer, "43% of Data Breaches Connected to Application Vulnerabilities: Assessing the AppSec Implications," Contrast Security Blog, May 20, 2020.

CONTRAST SECURITY

240 3rd Street
Los Altos, CA 94022
888.371.1333

Contrast Security is the world's leading provider of security technology that enables software applications to protect themselves against cyberattacks, heralding the new era of self-protecting software. Contrast's patented deep security instrumentation is the breakthrough technology that enables highly accurate assessment and always-on protection of an entire application portfolio, without disruptive scanning or expensive security experts. Only Contrast has sensors that work actively inside applications to uncover vulnerabilities, prevent data breaches, and secure the entire enterprise from development, to operations, to production.