VOLUME 3

State of Software Security Report

The Intractable Problem of Insecure Software

April 19, 2011

VERACODE

LEADING UP TO THE PUBLISHING OF THIS REPORT, several large global organizations suffered major security breaches, one called the largest in history, at the hands of targeted, persistent, yet not entirely sophisticated attacks. Whether it's a spear phishing attack that hamstrung Epsilon and RSA, or hackers taking advantage of common SQL injection errors in the case of HBGary and the "Night Dragon" cyber-attacks targeting Western oil, gas and petrochemical companies, these are critical reminders about the vulnerabilities within the core software infrastructure of organizations we rely on to communicate with customers, conduct financial transactions, supply power or store and share sensitive information. The related headlines remind us that software is ubiquitous, runs every part of our business and is sourced from all across the software supply chain. Your software supplier's weaknesses are your weaknesses.

These incidents, despite being classified as "alarmingly simple" by some, are not surprising if you consider the statistics that Veracode has reported since we started publishing the State of Software Security reports. Now in its third Volume, the reports continue to explore the reasons behind the headlines, putting a spotlight on commonly occurring vulnerabilities, and more importantly, what can be done to stem the tide. We dig into the DNA of software applications and examine them from numerous angles—supplier type, language, platform and industry. Our goal is to inform, enlighten and in some cases delight our readers with real-world evidence that can make their organization more secure, at its core.

This volume captures data collected over the past 18 months from the analysis of 4,835 applications on our cloud platform (compared to 2,922 in Volume 2 published in September 2010). The fact that the number of applications has nearly doubled indicates that organizations are diligently working their way through their application portfolio to understand their security weaknesses. What's more is that they are addressing these issues in a responsible and expedient manner. We took a deep dive into the software sector and made some surprising findings about the quality of security and customer-focused vendor applications. We observed industries across the board including Financial, Software & IT Services and Aerospace and Defense increasingly holding their software suppliers accountable for the security quality of the applications they are procuring from them. The reliance on third-party software applications is finally yielding to an independent assessment of their potential risks. And, we have new statistical evidence to demonstrate the business criticality of investing in greater application security training and education for development staff within existing application security programs.

As you examine the information presented here, I welcome your questions and ideas about what we can do collectively to accelerate the movement toward more secure software—more secure software to communicate with our customers, run our businesses and support the global economy.

Best Regards,

Matthew Moynahan
Chief Executive Officer, Veracode

Table of Contents

Introduction 2
Executive Summary 3
Software Supply Chain 7
Remediation Analysis 11
Third-party Risk Assessments 18
Security of Applications 22
Software Industry Analysis 36
Developer Training and Education 41
Application Threat Space Trends 43
Addendum 45
Assurance Level Definitions 46

Introduction

The State of Software Security is a semi-annual report that draws on continuously updated information in Veracode's cloud-based application risk management services platform. Unlike a survey, the data comes from actual code-level analysis of billions of lines of code and thousands of applications.

The resulting security intelligence cannot be found anywhere else. It represents multiple testing methodologies (static binary, dynamic, and manual) on the full spectrum of application types (components, shared libraries, web and non-web applications) and programming languages (including Java, C/C++, .NET, ColdFusion, and PHP) from every part of the software supply chain (Internally Developed, Open Source, Outsourced, Commercial). For those executives, security practitioners and developers who want to better understand the vulnerabilities that threaten the integrity and performance of the software supply chain, this series of reports is essential reading.

This volume captures data collected over the past 18 months from the analysis of 4,835 applications on our cloud platform (compared to 2,922 in Volume 2 published in September 2010). This reflects the growing use of independent, cloud-based application security testing services. As before, the report first examines the security quality of applications by supplier type in the software supply chain and then explores application security by language, industry, and application type.

New in Volume 3 are sections on Remediation Analysis, Developer Training and Education, and a deep dive on the Software industry.

Veracode welcomes any questions or comments from readers and will continually strive to improve and enrich the quality and detail of our analysis.

Executive Summary

The following are some of the most significant findings in the Veracode State of Software Security Report, Volume 3, representing 4,835 applications assessed in the last 18 months by Veracode on our cloud-based application security platform.

1. When first tested, more than half of all applications fail to meet acceptable security quality, and more than 8 out of 10 web applications fail OWASP Top 10
2. Cross-site scripting prevalence remains constant over time, while SQL injection is trending slightly down
3. Finance and Software industries lead the charge on holding software suppliers accountable; Aerospace and Defense are following suit
4. Most developers are in dire need of additional application security training and knowledge
5. The Software industry, including security products and services, have significant gaps in their security posture
6. While static analysis finds orders of magnitude more flaws than dynamic analysis, both techniques are required for comprehensive coverage
7. Building secure software or requiring it from your suppliers does not have to be time consuming

Key Findings

1. When first tested, more than half of all applications fail to meet acceptable security quality, and more than 8 out of 10 web applications fail OWASP Top 10

58% of all applications were deemed to have "unacceptable" security quality upon first submission (Figure 3). This remains essentially unchanged from the statistic that was reported in Volume 2 (57% unacceptable).[1] Commercial acceptability dipped a small amount from 35% acceptable in Volume 2 to 32% in this Volume. When measured against the OWASP Top 10, an industry standard list of critical web application errors, more than 8 out of 10 web applications across internally developed and commercial supplier types fail to achieve compliance (Figure 10). OWASP Top 10 is one of the standards relied upon by the PCI council so this failure rate also gives one insight into the poor state of non-compliance with respect to regulations such as PCI. This poor state of security of applications on their first submission to Veracode is due to two possible factors; either security processes such as threat modeling or secure coding standards were not incorporated into the development lifecycle, or the security processes were incorporated but failed to reduce flaws significantly. When you consider these statistics in the context of the ever strengthening threat environment these application security weaknesses translate into real and present danger for the risk-free operation of your software infrastructure. The 2010 Verizon Data Breach Investigations Report estimates that 40% of breaches occur due to hacking (i.e. successful exploitation of a software vulnerability) and are responsible for 96% of the compromised records.[2]

Recommendation: More training and more testing. It is clear that there is room for significantly greater emphasis on training and awareness of common security vulnerabilities such as the OWASP Top 10 and CWE/SANS Top 25. The training should be reinforced with consistent testing for compliance with these benchmarks for both internally developed and third-party applications.

[1] http://info.veracode.com/State-of-Software-Security-Volume-2.html

[2] www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

2. Cross-site scripting prevalence remains constant over time, while SQL injection is trending slightly down

This report examined trends in Cross-site scripting (XSS) and SQL injection, the two most commonly discussed issues in web application security, by looking at the percent of applications affected, quarter over quarter since the beginning of 2009. We see XSS remaining nearly flat and SQL injection gradually decreasing by 2.4% per quarter, a statistically significant amount according to linear regression (Figure 20, Figure 21). On one hand, it's reassuring that the industry seems to be making progress at reducing SQL injection; on the other hand, it's disappointing that we're not seeing a steeper downward trend. This is particularly concerning given that XSS and SQL injection are hot-button issues in many enterprises and ones that most organizations are actively trying to reduce. This trend could indicate that testing and remediation efforts are just barely keeping pace with development of new applications. Perhaps the most alarming realization is that when you consider the threat environment, these vulnerabilities become more than just theoretical flaws in your code base. They are ticking time bombs waiting to be exploited in real world attacks. The 2010 Data Breach Investigations Report from Verizon reveals that 25% of attacks carried out via hacking techniques were attributable to SQL injection and 8% were attributable to XSS.[3]

Recommendation: SQL injection and XSS are repeatedly in the headlines as the initial attack vector for high-profile, targeted breaches. It is crucial to reduce their occurrence in software applications if we are to outpace attackers. Organizations are encouraged to double down on their efforts to train their development and security staff on how to avoid these errors in the first place or fix them quickly once they are found. Use automated testing techniques to expediently discover these vulnerabilities across your application portfolio and to verify that the development team is following the guidance learned from their training. There are even free services such as Veracode's free XSS detection service that can provide development teams a quick view into the XSS issues present in their web applications.[4]

3. Finance and Software industries lead the charge on holding software suppliers accountable; Aerospace and Defense are following suit

One of the fastest growing areas within application security is independent verification of third-party software. As organizations are breached because of vulnerabilities present in someone else's software, they are starting to hold their software suppliers more accountable. Organizations are demanding proof of independent security verification before proceeding with a commercial transaction. The two industry segments leading the charge in this movement are Finance and Software. Together they represent over 75% of the enterprises requesting formal verification of third-party suppliers (Figure 12). It was interesting to see Aerospace and Defense, an industry that prides itself in the rigor it brings to its manufacturing supply chain, starting to apply the same standards to its software supply chain. The results of these independent assessments are enlightening: 25% of third-party applications are found to be of acceptable security quality upon initial submission (Figure 15). While this marks a slight improvement from Volume 2 (19% acceptable) clearly most software suppliers have significant work to do to ensure they are complying with the security gate set by the purchasing enterprise.

Recommendation: Reliance on third-party software to perform critical business functions and collaboration amongst an organization's workforce is only going to increase with the adoption of cloud and mobile platforms. Not having visibility into the security of these third-party applications is leaving a blind spot in an organization's understanding of its risk posture while providing yet another attack point for malicious parties. Maintaining the status quo is simply not an option. Software purchasers should introduce independent security verification language into their legal contracts and require proof of independent testing as part of their procurement process. Software producers should participate in this process in a cooperative and transparent manner as it ultimately serves to elevate the security posture of their product. This in turn can be used as a competitive differentiator in the marketplace.

[3] www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

[4] www.veracode.com/freeservice

4. Most developers are in dire need of additional application security training and knowledge

Over 50% of users taking an application security fundamentals exam received a grade of C or lower. Over 30% received a failing grade of D or F (Figure 31). The exam covers knowledge of broad security concepts, including common threats, and may be taken by developers, managers, or QA testers. Considering these exam scores, it is no wonder that over 50% of applications fail to achieve acceptable security quality upon initial submission. Performance on other exams such as Secure coding for Java, Secure coding for .NET and Introduction to Cryptography didn't fare much better. Anywhere from 35% to 48% of users taking those courses received a grade of C or lower (Figure 31).

Recommendation: Application security training and education is not a formal part of most computer science curriculums and certainly not a consistent theme in the professional development opportunities made available to technology professionals in companies. Therefore the results obtained from these exams are no surprise. Organizations are strongly encouraged to institute developer training and education programs to ensure a high competency level on application security. Take advantage of eLearning platforms to provide this training in a cost-effective and scalable manner. Close the loop on training by allowing developers to test their code using automated analysis techniques.

5. The Software industry, including security products and services, have significant gaps in their security posture

Similar to our deeper analysis of the financial industry in Volume 2, we engaged in a deep dive on the software industry segment in this report. What we found was both surprising and disappointing. It also served to explain the recent breaches that have been carried out against prominent security vendors such as RSA, HBGary and Comodo. Overall, 66% of Software industry applications were found to be of unacceptable security quality upon initial submission (Figure 28), which is worse than the 58% unacceptable rate when applications from all industries are taken into account. When measured against Veracode's risk adjusted verification methodology the two worst performers within the Software industry were the sub-categories of customer support (82% unacceptable) and most surprisingly security products and services (72% unacceptable) (Figure 28). The customer support category includes customer relationship management and web customer support applications. The security products and services category includes applications that are being used to perform a security function. The good news was that overall the Software industry demonstrated the ability to meet acceptable security quality in a timely manner. Over 90% of all applications across the Software industry achieved acceptable security quality within 1 month. The average for all applications in the security products and services sub-category was an impressive 3 days.

Recommendation: There are a few important lessons to be learned from this analysis. Security should not be assumed on the part of any industry segment even when it is those producing software for a living—including security software. A formal independent verification of security quality must be mandated in the procurement process as well as in the SDLC before pushing applications to production. Further, it should be noted that the verification process does not have to slow down the software acquisition or software deployment timeline. The data confirms that acceptable security policy can be achieved within reasonable timeframes. Compare the test results of different applications developed by staff with varying levels of security training to understand where training is working and to fill in gaps.

6. While static analysis finds orders of magnitude more flaws than dynamic analysis, both techniques are required for comprehensive coverage

Static analysis was able to find significantly more flaws as dynamic analysis in important categories such as XSS, CRLF injection and SQL injection (overall as much as 22 times more) (Table 5). One major contributing factor in the volumes of flaws found is that static analysis provides comprehensive coverage of the application whereas dynamic analysis only tests code paths that it can discover externally. Often, dynamic (and even manual) testing completely overlook portions of the application that are only reachable under certain circumstances (e.g. functionality that is gated behind a series of forms that trigger different behavior depending on how they are filled out). On the other hand, all the static findings will not necessarily be exploitable; dynamic and manual analysis are better at determining exploitability.

Recommendation: The lesson for CISOs and CIOs is that a robust application security program must incorporate multiple testing methods in order to ensure that applications are assessed with sufficient coverage, measured by both depth and breadth. Becoming overly dependent on too few analysis methodologies guarantees blind spots when assessing overall application risk. When you consider that prominent breaches such as Heartland Payment Systems, HBGary, and the "Night Dragon" cyber attacks targeted at companies like Shell, Exxon Mobil and BP, had SQL injection as their root cause, it's prudent to maximize your chances of finding as many instances of this issue in your code base as possible. Combining multiple testing techniques allows you to do exactly that. If your testing process is currently limited to automated dynamic and or manual testing adding static testing can greatly improve flaw identification.

7. Building secure software or requiring it from your suppliers does not have to be time consuming

Over 50% of commercial suppliers in our dataset resubmitted 90-100% of their applications and slightly under 40% of companies developing applications internally resubmitted 90-100% of their applications (Figure 4). When all applications were measured against Veracode's risk adjusted verification methodology, it was found that more than 80% across all supplier types achieved an acceptable security quality within 1 month (Figure 7). What this tells us is that when developers and security professionals attempt to do the right thing (i.e. achieve the requisite security rating), they can do so quickly and efficiently. The trick is to have the right application security training, testing tools and accurate guidance on where the vulnerabilities are and how to fix them. No one intends to write insecure code, so when they are trained appropriately and made aware of the security weaknesses that exist in their work products, they can act on that information and strive to achieve acceptable security quality expediently.

Recommendation: CIOs and CISOs should take relief in the knowledge that when the right application security training, technologies for security verification and guidance on security weaknesses present in their applications are made available to their development staff they will take responsibility and pursue the appropriate corrective actions expediently. The same is true for an organization's software suppliers. Arm your teams with the right training to avoid mistakes in the first place, but equally as important, implement a formal application security program for internally developed and third-party applications to improve the state of software security in your organization.

Software Supply Chain

There is one fact about the composition of applications that remains unchanged report after report—No application is written entirely from scratch and no company's application portfolio is free of third-party applications and code components. If anything we are seeing a greater reliance on third-party applications for performing mission critical operations.

In this section we examine the security quality of software produced by the software supply chain most often found in organizations: Internally Developed, Commercial, Open Source, and Outsourced. Only by understanding the various degrees of software security quality produced by supply chain participants can we begin to understand the requirements to change policies and processes, properly manage application risk in organizations, and protect critical software infrastructure. We also explore the fast growing third-party risk assessment market to determine the dynamics at play and what software producers and software purchasers are learning from engaging in a transparent and independent security verification process.

There is one undeniable trend. Software security is no longer assumed. Proof of independent verification is being sought by CIOs and CISOs as part of the procurement or outsourcing process.

There is one undeniable trend. Software security is no longer assumed. Proof of independent verification is being sought by CIOs and CISOs as part of the procurement or outsourcing process.

Veracode sampling continues to find that between 30 and 70% of code submitted as Internally Developed is identifiably from third-parties, most often in the form of Open Source components and Commercial shared libraries and components.

Distribution of Application Development by Supplier Type

Figure 1 reveals similar statistics to Volume 2 of the report. Close to a third of the applications analyzed during the reporting period were identified as third-party (Commercial, Open Source and Outsourced vendors). The percentage of outsourced applications represented in the dataset remains low at 1%. We continue to believe that part of this is a data labeling issue. Organizations sometimes consider code developed by outsourcers as "internally developed." Veracode encountered many instances where flaws in "internally developed" code were traced back to software supplied by outsourcing partners. Another factor is that outsourcing contracts have been silent on the topic of security testing and remediation. As these contracts renew, Veracode expects to see independent security verification requirements inserted and an increase in the percentage of identifiably outsourced code submitted.

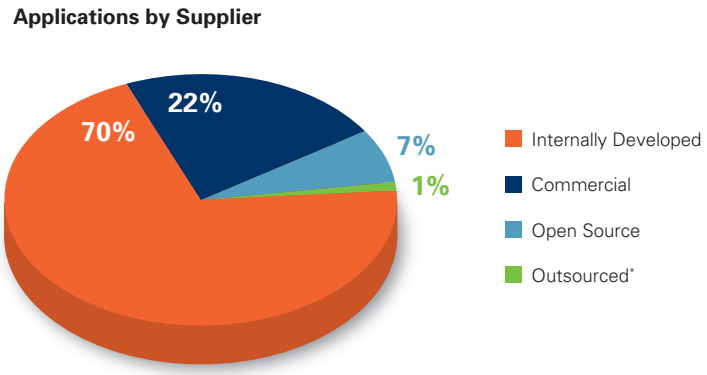


Figure 1: Applications by Supplier
(**Small sample size)*

Distribution of Application Business Criticality by Supplier Type

We know that there is a reliance on the extended software supply chain for procuring certain applications in any organization's application portfolio. As with the previous volume we explored whether the business criticality level of an application had a bearing on the choice of software supplier. This is depicted in Figure 2. Interestingly the percentage of applications designated as "Very High" business criticality that are being sourced from commercial suppliers has increased from 17% in Volume 2 to 29% in Volume 3. Percentage of "High" business criticality applications being sourced from commercial vendors remains unaltered at 25%. It remains true then that domain expertise, proven functionality, and time-to-market are all greater driving factors in the choice of software supplier as opposed to the business criticality. This only increases the importance of applying uniform application security verification processes across all supplier types. The section on third-party risk assessments later in the report discusses the results from independent assessments performed by Veracode in accordance with such policies by enterprises.

Increasing percentage of "Very High" business criticality applications being procured from commercial vendors (29% in Volume 3 as compared to 17% in Volume 2). This increases the importance of applying uniform application security verification processes across all supplier types.

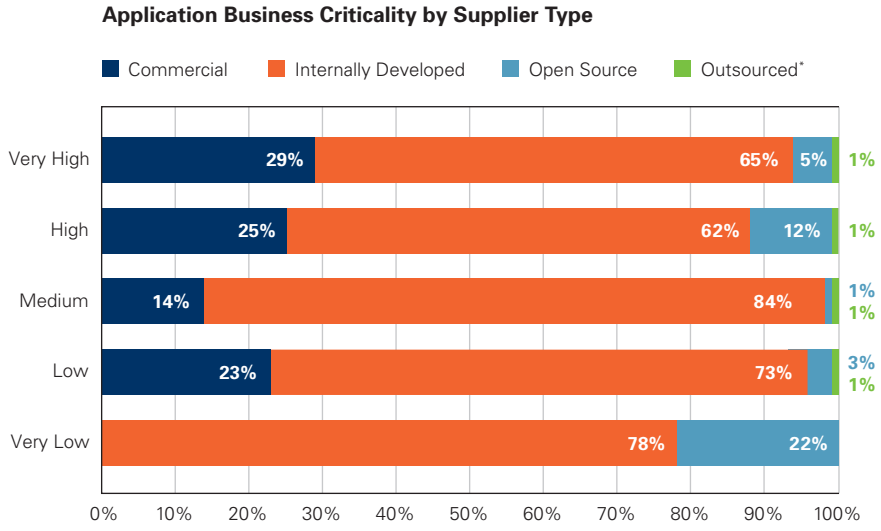


Figure 2: Application Business Criticality by Supplier Type
(**Small sample size)*

Distribution of Application Type and Programming Language by Supplier Type

Application Type and Programming Language by Supplier Type

	C/C++	ColdFusion	Java	.NET	PHP	Web	Non-Web
Internally Developed	7%	1%	56%	31%	5%	78%	22%
Commercial	19%	3%	49%	23%	6%	63%	37%
Open Source	25%	0%	59%	5%	11%	46%	55%
Outsourced*	4%	2%	78%	10%	6%	94%	6%

Table 1: Application Type and Programming Language by Supplier Type
(**Small sample size)*

New in Volume 3 is the inclusion of ColdFusion and PHP. As Table 1 shows, PHP is reasonably prevalent across all software suppliers. ColdFusion found in some use across Commercial, Internally Developed and Outsourced suppliers. There was a greater presence of web applications found across the software suppliers which is generally a reflection of the population of the overall dataset in favor of web applications. The diversity in languages and platforms across suppliers necessitates the importance of having a broad application security policy and testing approach that can handle disparate application types. For example, support for C/C++ and non-web applications is required when choosing security testing approaches for third-party software.

Support for C/C++ and non-web applications is required when choosing security testing approaches for third-party software.

Application Security Performance Across Supplier Types

Figure 3 depicts supplier performance on first submission as measured by the Veracode risk adjusted verification methodology (refer to addendum for detailed methodology). When calculated as a percentage of total applications submitted 58% of all applications were deemed to have "unacceptable" security quality upon first submission. This remains essentially unchanged from the statistic that was reported in Volume 2 (57% unacceptable). Commercial acceptance rate dipped a small amount from 35% acceptable in Volume 2 to 32% in Volume 3. It remains clear that all participants in the software supply chain still have a ways to go to improve the overall state of software security.

Overall acceptance rate of software across supplier types remains unchanged at 58% unacceptable (57% in Volume 2). Commercial acceptance rate dipped a small amount from 35% to 32%.

This poor state of security of applications on their first submission to Veracode is due to two possible factors; either the development team did not perform any application security processes during development, such as threat modeling or secure coding standards, or the security processes were performed but failed to reduce flaws significantly.

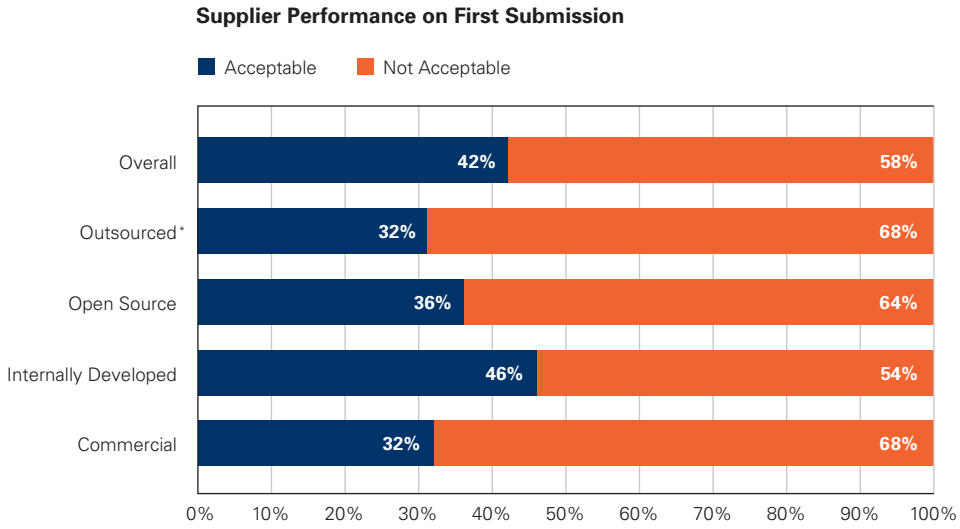


Figure 3: Supplier Performance on First Submission
(**Small sample size)*

Remediation Analysis

Ever since we first started publishing the State of Software Security report in early 2010, we have received a lot of interest in understanding the remediation workflow performed by organizations. Questions such as, what percentage of applications and vulnerabilities get remediated, how long does it take, what is the improvement in the security quality, are all frequently raised queries by customers and analysts. Therefore in this report, we decided to dedicate a section to remediation analysis. This section attempts to unravel the process that organizations follow once vulnerabilities have been reported to them as a result of some kind of application security test (static, dynamic or manual).

Percentage of Applications Resubmitted by Supplier Type

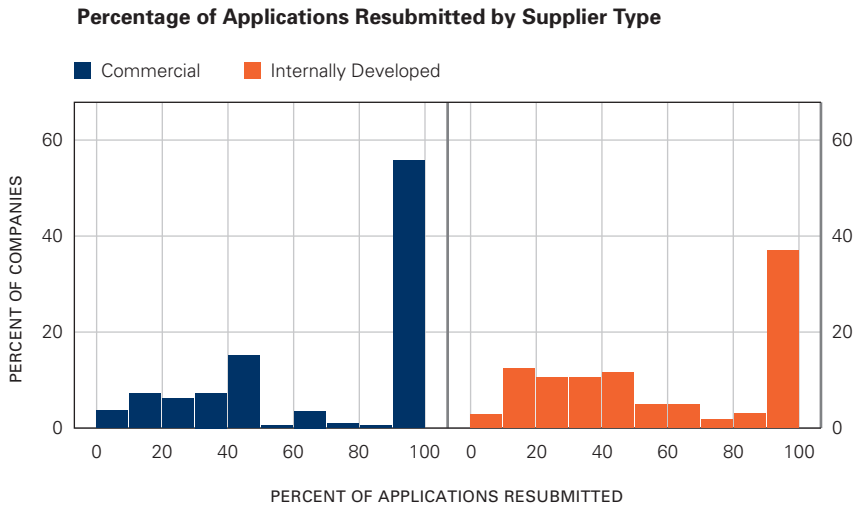


Figure 4: Percentage of Applications Resubmitted by Supplier Type

Figure 4 presents a graph that examines how frequently companies resubmit builds of their commercial or internally developed applications following the initial analysis. These resubmitted builds typically contain a combination of security fixes for previously reported vulnerabilities as well as new or altered code components that are not a result of security fixes and may represent new functionality. In some instances, particularly for commercial software, the resubmission may be characterized as a whole new version of a market facing release of the product in question. The graph presents histograms for two categories of suppliers: Commercial and Internally Developed. Each bar shows what percent of submitting companies resubmit some percentage (0-10%, 10-20%, … 90-100%) of their applications. An application is considered resubmitted if it is analyzed at least twice by the same technique—static, dynamic, or manual. As the two histograms show, over 50% of companies in our dataset resubmitted 90-100% of the commercial applications and slightly under 40% of companies in our dataset resubmitted 90-100% of their internally developed applications. It is important to interpret this figure correctly. While it might seem to be a positive indicator when a high percentage of companies are resubmitting 90-100%, of their applications, it is actually mixed.

On the one hand, it is good when companies resubmit applications that do not achieve adequate security quality on the first try. Indeed, this indicates serious diligence in pursuit of acceptable security quality. On the other hand, a high resubmission rate indicates a need to resubmit (failure to achieve acceptable quality on at least one review) which is not good. Similarly, if a high percentage of companies are resubmitting only 0-10% of their applications, this could mean that applications do not need to be resubmitted (good) or that companies are not diligent about resubmission (bad). The bottom line is that the above histograms suggest that a reasonable portion of companies (approximately 50% for commercially supplied applications and 40% for internally developed applications) are being diligent by resubmitting 90-100% of the applications that need improvement.

Over 50% of companies submitting commercial applications resubmitted 90-100% of them at least once. Over 40% of companies submitting internally developed applications resubmitted 90-100% of them at least once. This indicates reasonable diligence but there is room for improvement.

Percentage of Applications Resubmitted by Industry Type

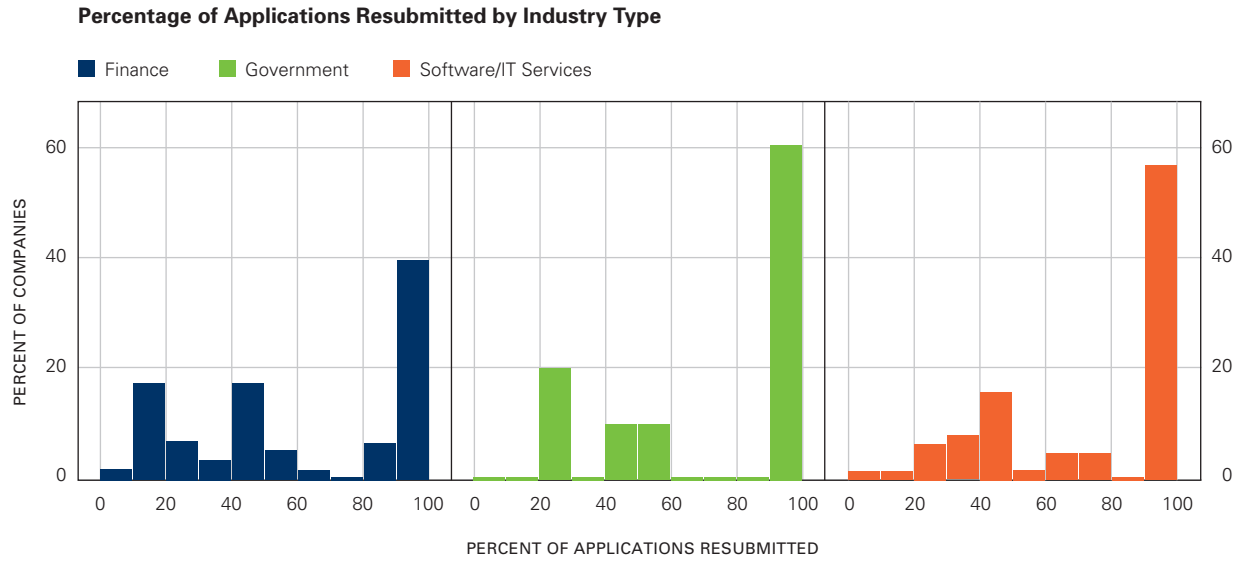


Figure 5: Percentage of Applications Resubmitted by Industry Type

Figure 5 shows a similar analysis to the previous figure, but this time broken down by Industry. Approximately 40% of companies in the Financial industry vertical resubmitted 90-100% of their applications. Government and Software companies had resubmission rates that were significantly higher at approximately 60% and 55% of companies resubmitting 90-100% of their applications. In general this shows moderate efforts across industry verticals to remedy the vulnerabilities in their software applications.

Resubmission rates indicate moderate remediation activities across industry verticals.

Percentage of Applications Resubmitted by Business Criticality

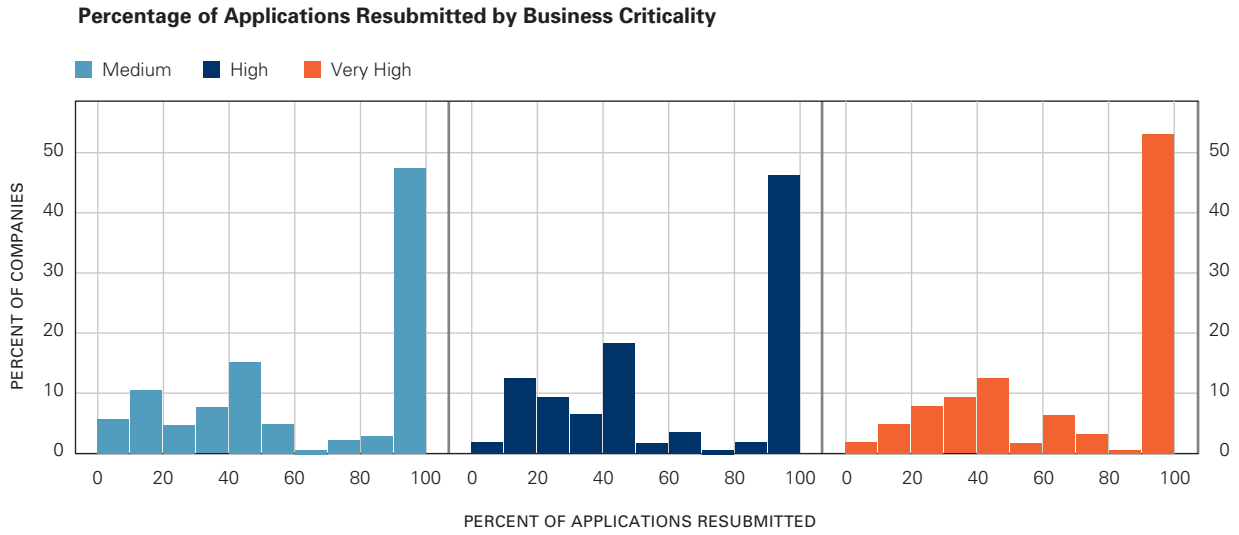
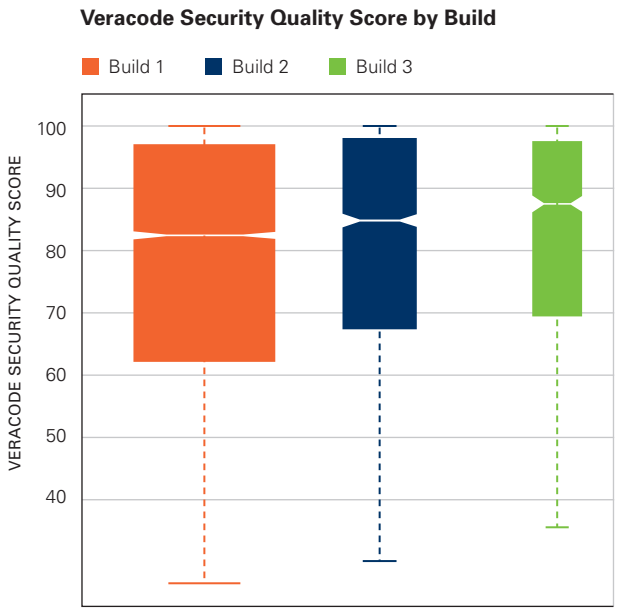


Figure 6: Percentage of Applications Resubmitted by Business Criticality

Next we performed the same analysis of resubmission activity broken down by application business criticality. The results are depicted in Figure 6. We expected to see higher resubmission activity with increased business criticality for three reasons. First, organizations often commence with a broad vulnerability discovery process that spans their application inventory and then prioritize remediation activities for their higher criticality applications. Second, the criteria for achieving acceptable security quality become more demanding as application critically increases. Third, the security quality is presumably more important as the business critically of applications increases. What is surprising is that only slightly more than 50% of companies resubmit 90-100% of their very high criticality applications. This is lower than we expected. Also, the difference in remediation activity for "High" and "Medium" criticality applications is not statistically significant. This is not consistent with our expectation that remediation activity for high criticality applications would be greater than for medium applications. This is something that we will be tracking as our sample size continues to grow. Our recommendation to customers is to perform vulnerability assessments across their entire application portfolio and use business criticality as one of the prioritization criteria for remediation activities. Automated vulnerability assessments can help accelerate the vulnerability discovery process across large application inventories.

It is surprising that only slightly more than 50% of companies resubmit 90-100% of their "Very High" business criticality applications. Our recommendation to customers is to perform vulnerability assessments across their entire application portfolio and use business criticality as one of the prioritization criteria for remediation activities. Automated vulnerability assessments can help accelerate the vulnerability discovery process across large application inventories.

Veracode Security Quality Score by Build



Resubmission activity does lead to higher security quality scores build over build.

Figure 7: Veracode Security Quality Score by Build

In addition to investigating resubmission activity, we wanted to look at whether or not the activity leads to more robust and secure software. Indeed it does, as depicted in Figure 7 above. This analysis compares the range of Veracode Security Quality Score (SQS) as achieved on the first, second, and third reviews. The non-overlapping notches (around the median SQS line at the narrowest width of the boxes) in the above whisker plot suggest a statistically significant increase in SQS from build one to two to three. The width of the whiskers depicts a depleting sample size of application build increases. This sample size depletion is due to the fact that a significant portion of applications that did not achieve security policy on the first try, do achieve it on the second. This is also true for the decrease in sample size between build 2 and build 3. Collectively, these results suggest that resubmission activity does lead to higher SQS and presumably more secure software.

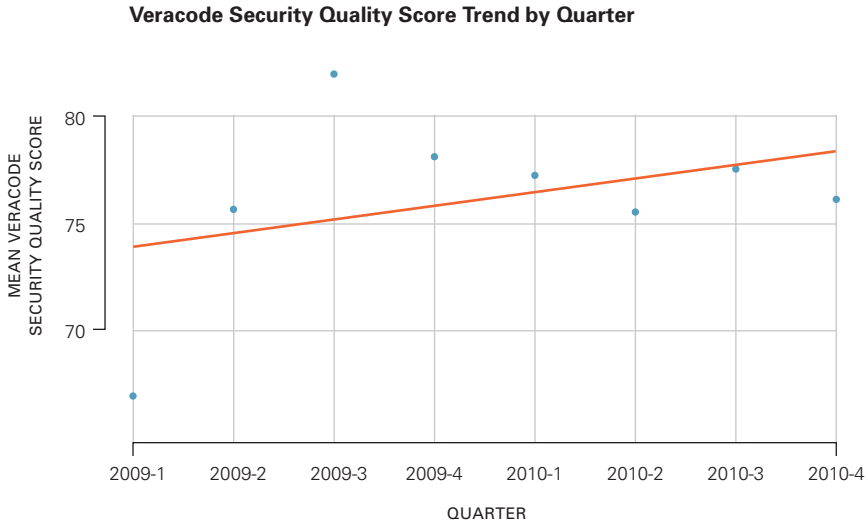


Figure 8: Veracode Security Quality Score Trend by Quarter

Figure 8 shows the average raw security quality score of applications, by quarter. Using a "best fit" linear regression, the trend seems to be a modest increase in security score over time. However, from a statistical perspective, the trend is flat. While it would have been encouraging to see a more marked improvement across the board, this trend may reflect the influx of new, previously untested applications into the mix. In other words, while many applications are in fact becoming more rugged, the average is being pulled down by new applications with lower security quality.

Using a "best fit" linear regression, the trend seems to be a modest increase in security score over time. However, from a statistical perspective, the trend is flat.

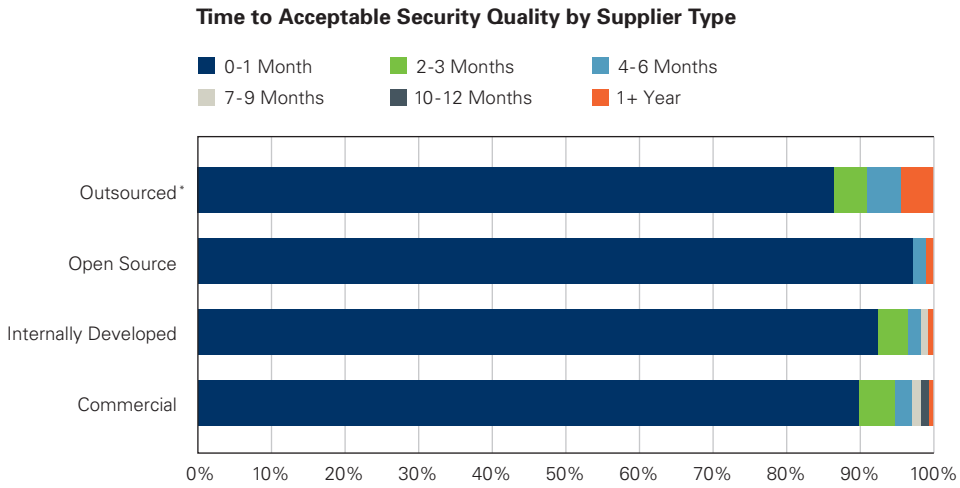


Figure 9: Time to Acceptable Security Quality by Supplier Type
(**Small sample size)*

The figures presented previously in this section explore the resubmission rate exhibited across various dimensions such as supplier type and business criticality. Generally the purpose of resubmitting remediated builds is to close the gap between the security quality of the application upon initial submission and the acceptability as dictated by business criticality. Figure 9 measures the time it takes get to acceptable security quality (as measured by Veracode's risk adjusted verification methodology). Instead of just reporting a mean number of days per supplier type, above we depict much more information in the form of a distribution of acceptable security quality achievement timeframes, ranging from 0 days (success on the first try) to over a year. Applications that were found to be of acceptable security quality upon initial submission are included in this dataset within the 0-1 month category. The overwhelming majority of applications across supplier types achieve acceptable security quality within the first month. This is encouraging to see as it indicates that once security and development professionals are made aware of security weaknesses in their applications they are quick to take action.

More than 80% of applications across supplier types achieve acceptable security quality within 1 month. May indicate that once security and development professionals are made aware of security weaknesses in their applications they are quick to take action.

Distribution by Ability to Meet Security Compliance Policy by Supplier

Figure 10 shows the percentage of web applications that met the OWASP Top 10 (2010) policy by supplier. An application was labeled "Not Acceptable" if it contained any vulnerabilities covered by the Top 10. The number of Commercial and Internally Developed web applications that were not acceptable remains high at more than 80%. This represents a modest improvement from our last report, in which 88% of Internally Developed and 93% of Commercial applications were unacceptable by OWASP Top Ten standards. While the trend is encouraging, the actual numbers are far from comforting.

One challenge that will continue to plague web applications is the pervasiveness of vulnerabilities such as Cross-site scripting (XSS). Because the presence of a single XSS flaw will designate an application as Not Acceptable by OWASP Top 10 policy, most web applications will have a difficult time reaching Acceptable status until developers focus more steadfastly on eradicating these common issues. Given the lack of improvement we've observed in XSS quarter over quarter (see Figure 20), it would be surprising to see the Acceptable percentage surpass 50% any time soon.

OWASP Top 10 Compliance by Supplier on First Submission
(Web Applications)

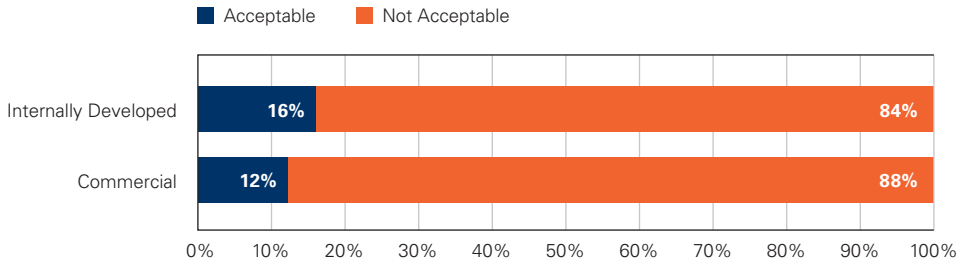


Figure 10: OWASP Top 10 Compliance by Supplier on First Submission (Web Applications)

Figure 11 examines suppliers' ability to deliver applications as measured by compliance against the CWE/SANS Top 25 Most Dangerous Software Errors. Only non-web applications were included in this analysis as web applications are more commonly measured against the OWASP Top 10, as shown in Figure 10. This is different from our last report, where CWE/SANS Top 25 compliance was reported across all applications. Internally Developed applications performed the best with 62% of applications meeting acceptance. Commercial applications fared slightly worse. This is unsurprising given that many enterprises are just beginning to factor the software supply chain into their software assurance programs where many of those applications may have never been independently tested before.

CWE/SANS Top 25 Compliance by Supplier on First Submission
(Non-Web Applications)

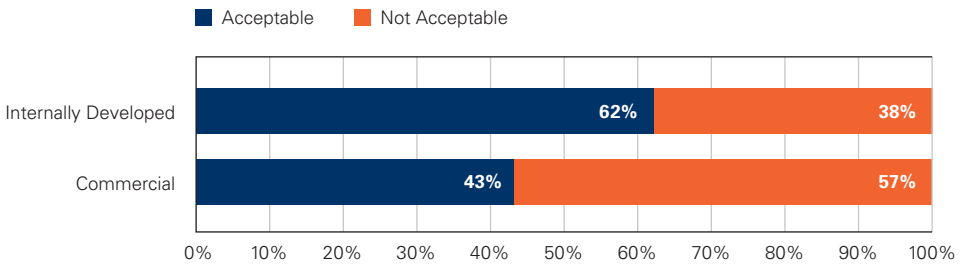


Figure 11: CWE/SANS Top 25 Compliance by Supplier on First Submission (Non-Web Applications)

Distribution of Most Common Security Vulnerabilities by Supplier

The distribution of security vulnerabilities by type of supplier may point to more or less effective practices and help in choosing future suppliers. Table 2 reveals relatively similar results by suppliers in terms of both prevalence and type of vulnerabilities detected. Considering all factors, including the contribution to types of vulnerabilities from choice of programming language and whether or not the application is a web application, the top vulnerability categories are similar across all suppliers.

Vulnerability Distribution by Supplier

Internally Developed		Commercial		Open Source		Outsourced*	
Cross-site Scripting (XSS)	52%	Cross-site Scripting (XSS)	47%	Cross-site Scripting (XSS)	36%	CRLF Injection	37%
CRLF Injection	13%	Information Leakage	14%	Information Leakage	14%	Cross-site Scripting (XSS)	37%
Information Leakage	13%	CRLF Injection	8%	Directory Traversal	13%	Information Leakage	8%
SQL Injection	4%	Cryptographic Issues	5%	CRLF Injection	12%	Encapsulation	6%
Cryptographic Issues	4%	Directory Traversal	5%	Cryptographic Issues	9%	Cryptographic Issues	3%
Directory Traversal	3%	Error Handling	4%	Time and State	3%	Credentials Mgmt	3%
Encapsulation	3%	Buffer Overflow	4%	Error Handling	3%	API Abuse	2%
Time and State	1%	Potential Backdoor	3%	SQL Injection	3%	Time and State	1%
Insufficient Input Validation	1%	SQL Injection	3%	API Abuse	2%	Directory Traversal	1%
Buffer Overflow	1%	Time and State	2%	Buffer Overflow	1%	SQL Injection	1%

Table 2: Vulnerability Distribution by Supplier
(**Small sample size)*

Third-party Risk Assessments

This section on Third-party Risk Assessments was originally introduced in Volume 2 and has been updated with the most recent information here. Third-party risk assessments are classified as independent security assessments that are performed on third-party software using multiple testing techniques at the request of a buyer of that software or software development service. These buyers may be purchasing already developed applications for internal use, applications to be developed by someone else and then used internally, or applications of either type to be re-distributed under an OEM, VAR, or other re-licensing arrangement. In some cases enterprises are performing these assessments as part of their M&A due diligence process.

This is one of the most rapidly developing parts of our business and has contributed substantially to the growth of the dataset that is included in this report. This section reveals some key characteristics of this fast growing third-party risk assessment market and establishes not just its viability but the real benefits that can be realized by both the software producer and the purchaser that engage in this process.

Requester Type by Industry

Figure 12 shows the types of organizations that are at the forefront of creating the third-party risk assessment market. These organizations are generally instituting a formal policy typically integrated into their procurement process that requires independent verification of the security quality of third-party software. Not surprisingly Software/IT Services (including software producers and providers of IT services and equipment) and Financial (including Banks, Insurance and Financial Services) organizations are at the top of the list. The drivers for these two industries may be somewhat different. The Financial sector is heavily regulated with serious monetary and brand related consequences if financial or customer data are compromised. Furthermore they recognize that it doesn't matter if the source of the breach was a piece of third-party software (e.g. widely deployed commercial desktop applications), the liability is still theirs. They wish to ensure that all participants in their supplier ecosystem are providing secure code. Similarly, the Software and IT services segment may be reusing code components from third-parties in products labeled with their name and needs to protect its overall brand. They wish to ensure that all participants in their platform ecosystem are providing secure code. New on the board this time is Aerospace and Defense. It is encouraging to see that the rigor that this industry applies to their manufacturing supply chain is finally being expanded to include its software supply chain as well.

The Financial and Software/IT Services industry segments were found to be leading the charge on formal security verification of third-party software suppliers. The Aerospace and Defense industry sector appears to be extending the due diligence performed on their manufacturing supply chain to their software supply chain as well.

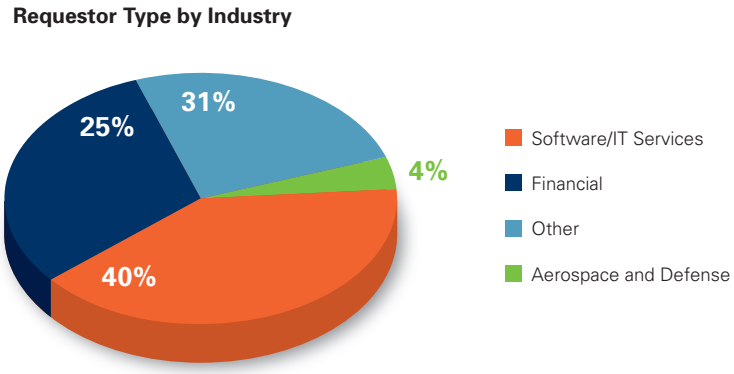


Figure 12: Requestor Type by Industry

Distribution of Third-party Assessments by Application Purpose

Next we examined the nature of the applications that were being analyzed by the above organizations. It appears that significant scrutiny is brought to bear by enterprises on third-party applications in the Operations and Financial categories. Typically these categories of third-party applications are transacting critical information (such as credit card or other financial information) or supporting critical business operations. It stands to reason then that an enterprise would want to assure themselves of the security of these third-party applications before acquiring them. Similar factors may be at play when you consider the next two largest categories of Learning and Growth and Customer Support. In the one instance sensitive employee information is at risk and in another sensitive customer data. Organizations that are selecting these types of applications for third-party assessments are clearly making application security a key element of their overall data protection strategy. It was surprising to see that the last category of Security Products and Services represented only 4% of the types of applications tested. Their infrequent selection could be because buyers likely assume these types of applications to be secure. This is a dangerous assumption to make and one that is shown not to be true later in the report (refer to Software Industry Analysis section).

The sensitivity of data and transactions are determining factors in the choice of third-party applications subjected to independent verification. However, security still only represented 4% of applications selected. As shown later in the report it is dangerous to assume that security products and services are themselves secure.

Third-party Assessments by Application Purpose

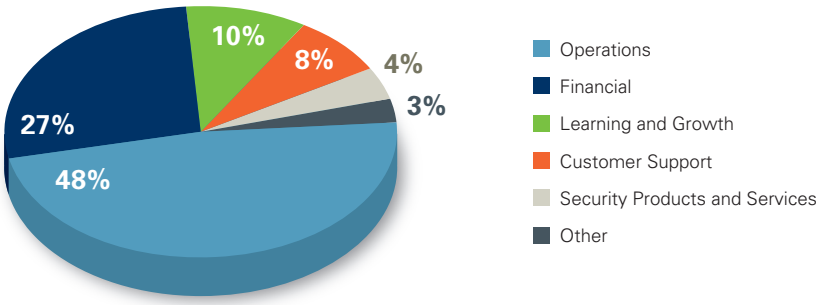


Figure 13: Third-party Assessments by Application Purpose

Application Type Definitions: *Operations category includes applications supporting day-to-day non-financial business activity such as product development, information management utilities, IT management tools etc.; Financial category traditional accounting and finance applications and newer mobile banking applications; Customer Support category includes customer relationship management and web customer support applications; Learning and Growth includes applications to support HR, training and human capital management. Security Products and Services indicate applications that are providing some type of security capability; other is represented by a combination of healthcare software, integration and other applications not included in a previous category.*

Figure 14 reveals that, in this reporting period there was a small improvement in the acceptability rate of third-party software upon initial submission. 25% of third-party applications were found to be acceptable upon initial submission (as opposed to 19% in Volume 2). While there is clearly a great deal of improvement that needs to occur in the security quality of third-party software (much like software in general) it is encouraging to see the statistics are moving in the right direction. One explanation may be that third-party software suppliers are becoming conditioned to their customers asking for independent verification of security quality prior to completing a commercial transaction. This in turn is prompting many of these vendors to integrate formal application security testing practices into their overall QA process resulting in improved scores when they are subjected to a third-party assessment.

The acceptability rate of third-party software upon initial submission showed a small improvement—25% in Volume 3 as compared to 19% in Volume 2.

It appears that third-party software suppliers are becoming conditioned to their customers asking for independent verification. Indeed, many choose to integrate formal application security testing into their development and QA processes proactively.

Third-party Assessments: Performance Upon Initial Submission

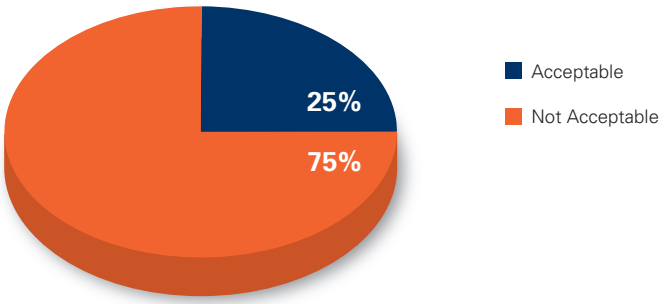


Figure 14: Third-party Assessments: Performance Upon Initial Submission

Security of Applications

The previous section presented information from the software supplier and purchaser perspectives in an attempt to help enterprises properly manage application risk in the software supply chain. In this section of the report we explore security risks related to web and non-web applications, programming languages, types of vulnerabilities, and industry alignment. New in this report, we provide a deeper investigation of application security within the software and IT services industry vertical.

As background, software vulnerabilities are the attack points in applications used by hackers to compromise a system. Different types of applications have different attack points. For example, web applications have different attack surfaces than desktop software or databases. Additionally, vulnerabilities can vary significantly by programming language and platforms such as the Windows versus Java operating systems. It is also possible for applications in different industries to have different vulnerabilities based on the secure coding skills of the engineering population serving those industries (e.g. Financial Services versus Retail) and the sophistication of their software development practices or central security teams.

While no software will ever be perfectly secure, understanding what makes applications more or less vulnerable provides the basis for CIOs, CISOs, and software professionals to manage application portfolio risk rather than remain blindly susceptible to breaches carried out by the successful exploits of zero-day and other application layer vulnerabilities.

Distribution of Applications by Type

All applications analyzed by Veracode are inventoried and classified according to a profile which includes key characteristics such as whether the application is web-facing, its language and platform, and the industry of the organization submitting it. Three-quarters of the applications analyzed were web applications (Figure 15). Obviously this represents the majority of the dataset. Some of this can be attributed to a heightened sense of risk associated with externally facing applications which has driven a higher submission rate for both internally developed and third-party web applications for security testing. Another contributing factor is that we have modified the manner in which we designate an application a web application to include not just information that was being provided by the submitter of the application but also characteristics that were discovered via automated analysis.

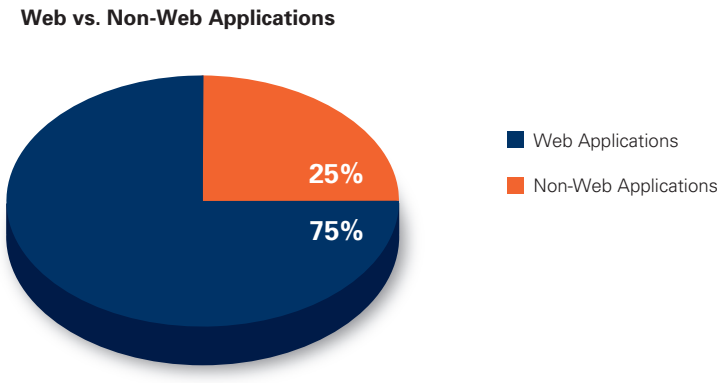


Figure 15: Web vs. Non-Web Applications

Distribution of Applications by Language

An analysis of the Distribution of Applications by Language is a useful indicator and reasonable proxy for the ever-changing attack surface of the world's software infrastructure. This is depicted in Figure 16.

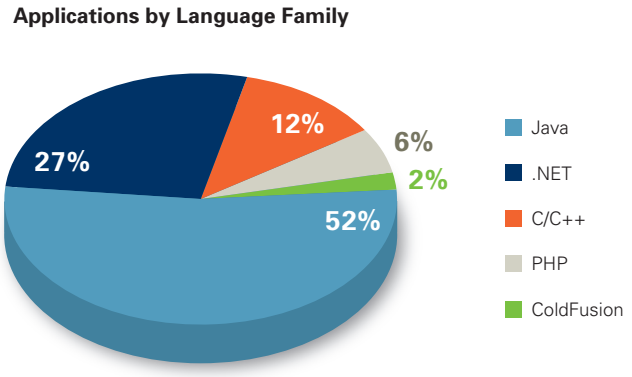


Figure 16: Applications by Language Family

In our last report we showed the relative distributions of five development platforms—Java, C/C++,.NET, PHP, and ColdFusion. The biggest change this time around is that C/C++ apps comprise only 12% of the data, down from 19%. An increase in PHP and ColdFusion accounts for this gap. Java and .NET still account for nearly 80% of all applications submitted for analysis, suggesting that many enterprises' application security strategy may not extend beyond public-facing web applications. This is also corroborated by the fact that three-quarters of our dataset represents web applications.

Flaw Density by Language

	First Quartile	Median	Mean	Third Quartile
C/C++	0.01	0.03	1.11	0.11
ColdFusion	1.59	2.60	7.59	10.89
Java	0.01	0.04	0.45	0.16
.NET	0.01	0.06	1.74	0.32
PHP	0.21	0.31	2.27	1.49

Table 3: Flaw Density by Language

To explore the impact of programming language on application security, Table 3 shows the median flaw density for each. The median flaws per thousand lines of code (KLOC) for Java, C/C++, and .NET are similar. Many people ask whether switching languages will improve application security.

Both ColdFusion and PHP exhibited much higher flaw densities, with medians of 2.6 and 0.3 flaws per KLOC, respectively. In our last report we also noted that ColdFusion's flaw density was unusually high. We speculated that this could be due to the compactness of the ColdFusion language as well as the sophistication of ColdFusion developers relative to Java or .NET developers. PHP could be explained similarly. While PHP code is not as compact as ColdFusion code, both languages are very easy to learn and encourage less disciplined coding practices than other languages.

We acknowledge that flaw density is an imperfect metric. For example, larger applications may contain higher proportions of framework code or interface classes, both of which inflate the KLOC count without actually adding functionality. Code comments are an additional source of inflation. Alternatively, we could measure flaws per megabyte of compiled code, but this would skew the metric differently, putting bytecode languages at an inherent disadvantage and producing different measurements based on compiler settings and optimizations.

Distribution of Applications by Vulnerability Type

The charts on the following pages depict top vulnerability categories by prevalence, presented from two different perspectives. The first is by vulnerability prevalence, which illustrates the percentage of the total vulnerabilities discovered. The second is by affected applications, which shows the percentage of applications containing one or more of the vulnerabilities in each category. Additionally, both vulnerability prevalence and affected applications are split into two views: web applications and non-web applications. Rows highlighted in red are vulnerability categories that also appear in the OWASP Top 10 (2010) standards and rows highlighted in green are vulnerability categories that appear in the CWE/SANS Top 25 (2010) standard. There is considerable overlap between Veracode findings and these industry standards, further confirming the relevance of these vulnerability categories as top areas of security weakness to focus on for enterprises.

In web applications, Cross-site Scripting (XSS) remains the most prevalent vulnerability category by frequency, accounting for 53% of all vulnerabilities in the data set (Figure 17). The next three categories behind XSS—Information Leakage, CRLF Injection, and Cryptographic Issues—maintain the same rankings as we observed in our last two reports.

Cross-site Scripting (XSS) remains the most prevalent vulnerability category by frequency, accounting for 53% of all vulnerabilities in web applications.

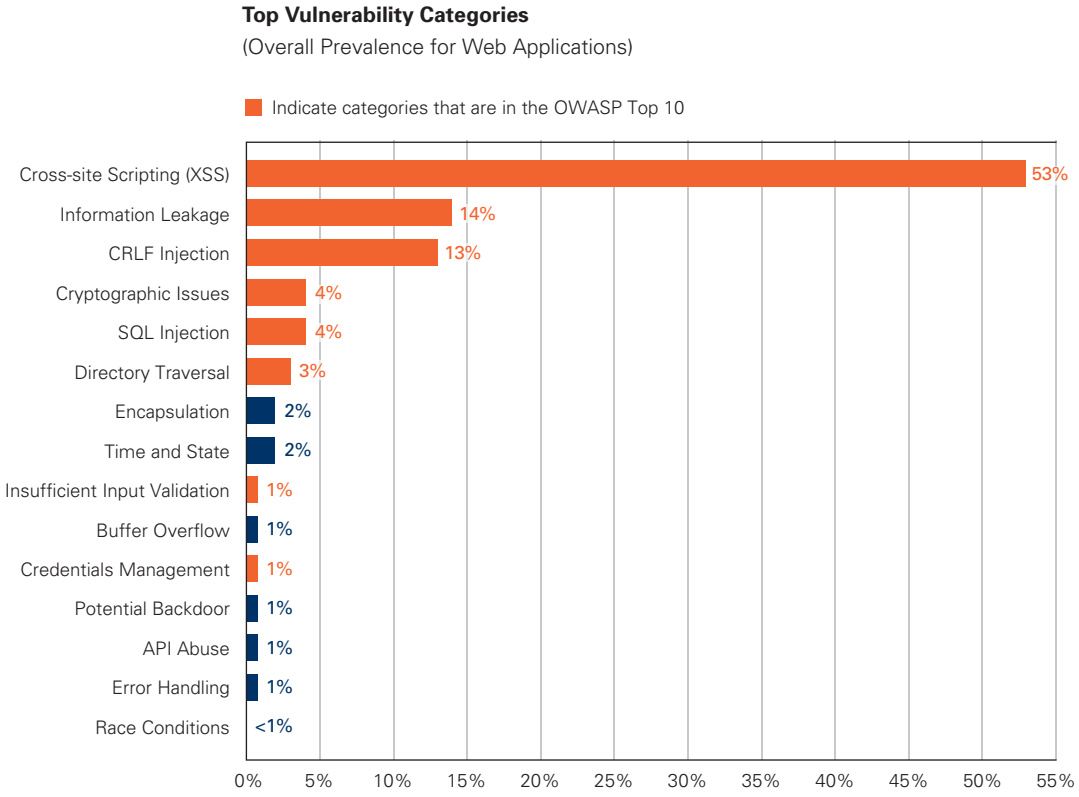


Figure 17: Top Vulnerability Categories (Overall Prevalence for Web Applications)

In non-web applications, buffer overflows and error handling account for over one-third of all vulnerabilities detected (Figure 18). This should not be surprising, as buffer overflows and related memory corruption vulnerabilities are notoriously difficult to eradicate due to the variety of ways in which they can manifest. Exacerbating the situation, many companies still have not banned the use of known dangerous functions (e.g. strcpy, strcat) despite the fact that safe alternatives have existed for years.

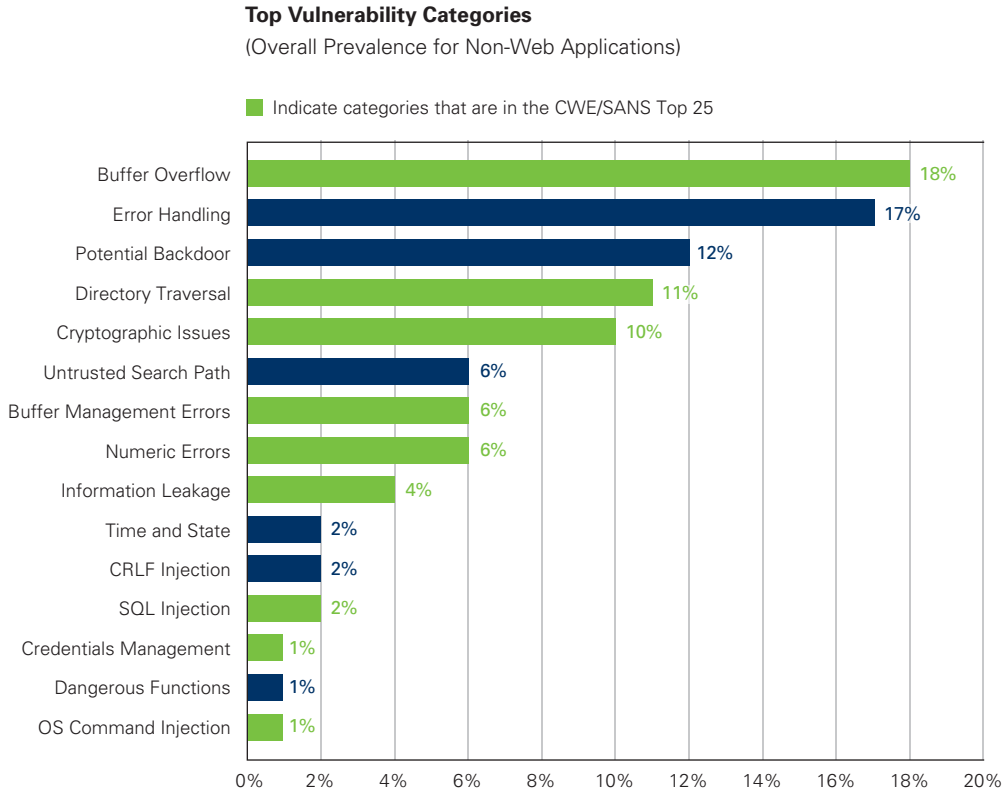


Figure 18: Top Vulnerability Categories (Overall Prevalence for Non-Web Applications)

Potential backdoors are detected surprisingly often in non-web applications, at 12% of all vulnerabilities. However, we remind readers that with respect to backdoor scans, automated scanning cannot reliably determine intent. For example, time bombs look the same as legitimate scheduled functionality. Aliasing functions or methods may look the same as call hiding. In order to distinguish the malicious cases from the legitimate ones, Potential Backdoors should be inspected carefully by someone with an understanding of the application's intended design and function.

Buffer overflows and error handling account for over one-third of all vulnerabilities detected in non-web applications. Potential backdoors comprise 12% of all vulnerabilities in non-web applications.

Even though a vulnerability category may account for a small percentage of the total vulnerabilities, the frequency with which it appears across different applications may be a more illuminating statistic. Viewing the vulnerabilities by affected web applications, Information Leakage tops the list, with over two-thirds of applications containing at least one vulnerability in this category (Figure 19). Information Leakage comprised flaws such as verbose error messages, disclosure of exception stack traces, and extraneous files on the server (detected dynamically), among others. Cross-site Scripting and Cryptographic Issues were also extremely widespread, appearing in 70% and 58% of all web applications, respectively.

Viewing the vulnerabilities by affected web applications, Information Leakage tops the list, with over two-thirds of applications containing at least one vulnerability in this category.

Top Vulnerability Categories
(Percent of Applications Affected for Web Applications)

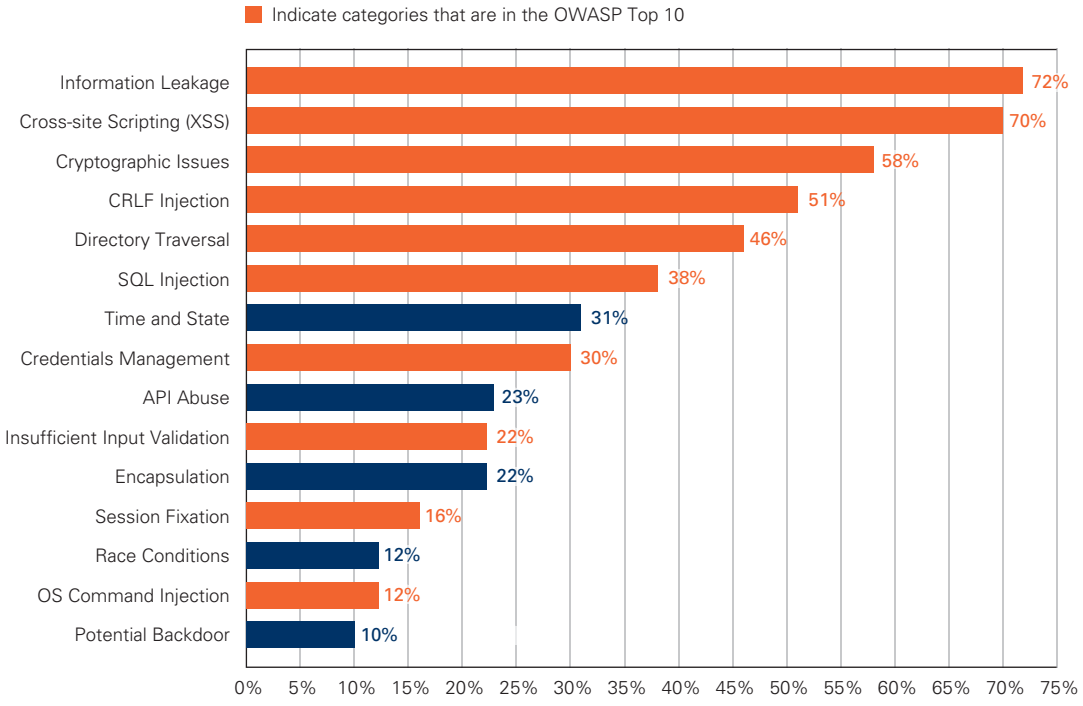


Figure 19: Top Vulnerability Categories (Percent of Applications Affected for Web Applications)

We also took a closer look at the trends in XSS and SQL Injection, the two most commonly discussed issues in web application security. Looking at the percent of applications affected, quarter over quarter since the beginning of 2009, we see XSS remaining nearly flat (Figure 20). Statistically speaking, the trend is flat. This is discouraging, particularly given that XSS is a hot-button issue in many enterprises and one that most organizations are actively trying to reduce. This trend could indicate that testing and remediation efforts are just barely keeping pace with development of new applications. It will be interesting to keep an eye on this trend over the coming years; the success or failure of XSS eradication efforts may be a bellwether for application security progress.

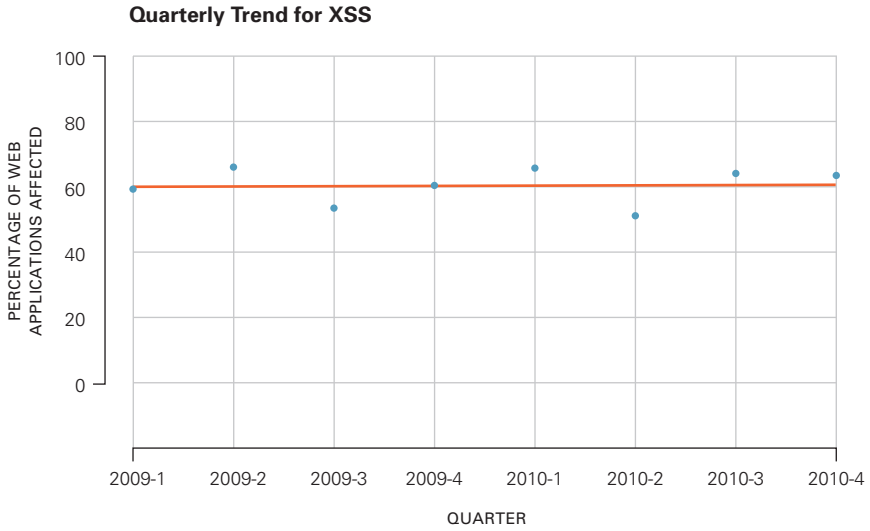


Figure 20: Quarterly Trend for XSS

The trend in SQL Injection is more promising. Over the past eight quarters, the percentage of applications affected by SQL Injection has gradually decreased by 2.4% per quarter, a statistically significant amount according to linear regression (Figure 21). One factor that may explain this progress is that SQL Injection is a very easy problem to understand and to fix. Unlike XSS where developers need to understand and apply context-specific encoding mechanisms, fixing SQL Injection is a matter of replacing ad-hoc database queries with parameterized prepared statements. There are some edge cases, but even those tend to be straightforward to fix. On one hand, it's reassuring that the industry seems to be making progress at reducing SQL Injection; on the other hand, it's disappointing that we're not seeing a steeper downward trend. SQL Injection has repeatedly been reported as the initial attack vector for high-profile, targeted breaches.

SQL Injection has gradually decreased by 2.4% per quarter while XSS is remaining flat.

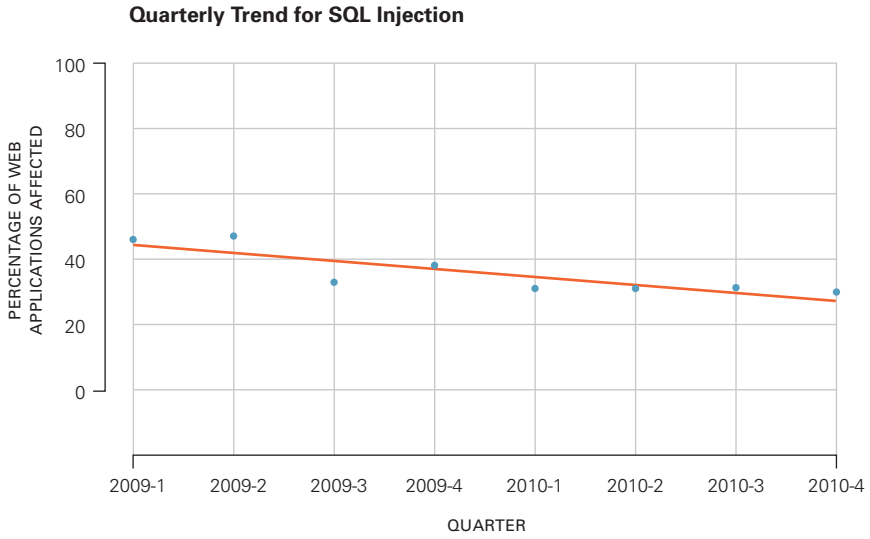


Figure 21: Quarterly Trend for SQL Injection

For non-web applications, Cryptographic Issues lead, with 53% of applications affected by at least one issue in this category (Figure 22). This category once again comprised insufficient entropy, plain text storage of sensitive data, use of hardcoded cryptographic keys, and use of algorithms with inadequate encryption strength. Directory Traversal and Error Handling round out the top three, at 36% and 28%, respectively. Directory Traversal may be surprising since it is a vulnerability class typically associated with web applications. However, in this data set, Directory Traversal is a generic term for path manipulation flaws, so for static analysis it describes any code construct where an untrusted value is used in the context of a file I/O operation.

For non-web applications, Cryptographic Issues lead, with 53% of applications affected by at least one issue in this category.

Top Vulnerability Categories
(Percentage of Applications Affected for Non-Web Applications)

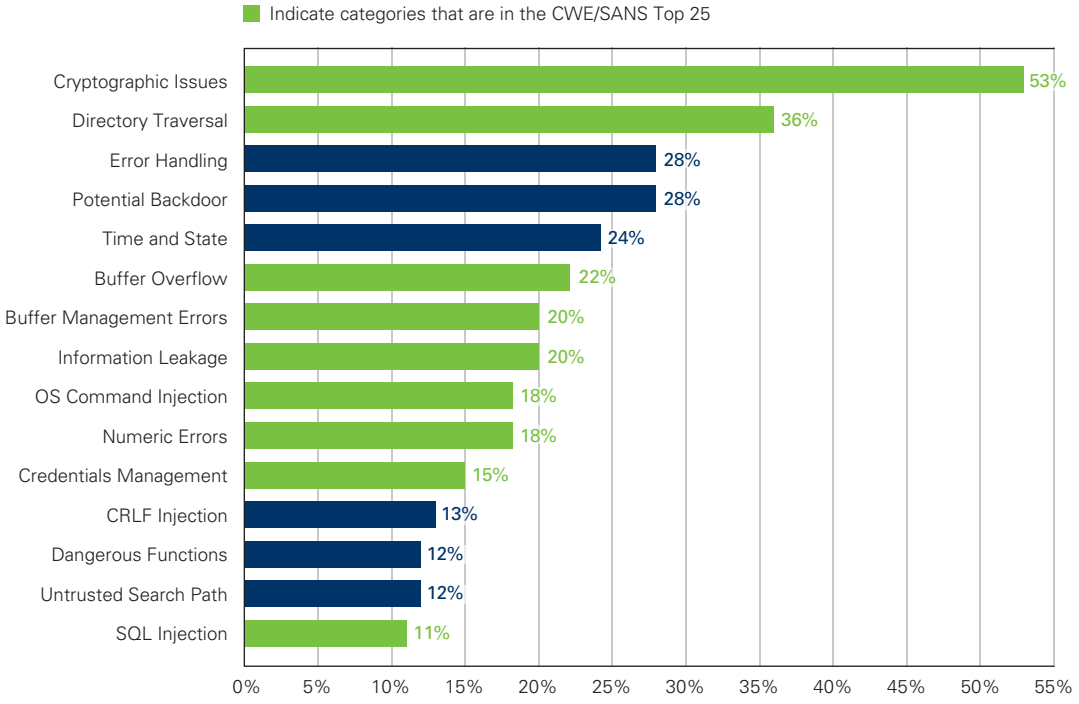


Figure 22: Top Vulnerability Categories (Percentage of Applications Affected for Non-Web Applications)

Vulnerabilities by Language Distribution

Table 4 presents the most prevalent categories (by share of total vulnerabilities discovered) based on language family. New to this volume of the State of Software Security report is the addition of data from the analysis of applications written in ColdFusion and PHP. These languages are used exclusively to write web applications so it is no surprise that the most common web application vulnerabilities rise to the top but in the case of ColdFusion and PHP there is clearly a heavy skew towards XSS vulnerabilities, much more so than Java and .NET. This is actually good news for developers using ColdFusion and PHP. They can significantly decrease the vulnerabilities in their applications by tackling the XSS category as a start to securing their applications.

Across the board we see XSS still dominating all the languages typically used to write web applications. Java and .NET are significantly lower than ColdFusion and PHP. This is likely because Java and .NET are used in more mature enterprise development environments where there is more awareness of the need to prevent XSS.

The other major web vulnerability that is the source of so many breaches that it makes the news almost daily is SQL Injection. This category doesn't make the top five for either Java or .NET but is ranked second for ColdFusion and third for PHP. This is another data point suggesting that more mature enterprise development teams have this common and dangerous vulnerability more under control with their development processes than teams using ColdFusion and PHP.

The rest of the top 5 most prevalent issues for web development languages share the same issues—Information Leakage, Directory Traversal, Cryptographic Issues, and OS Command Injection—but there is one standout that is unique to Java: CRLF Injection. Java applications tend to log much more than applications written in other languages. Logging is one of the many functions that is susceptible to CRLF Injection.

The non-type safety of C/C++ brings three of the top five to the list: Buffer Overflow, Numeric Errors (integer overflow and others), and Buffer Mgmt Errors. Error Handling is also more difficult in C/C++ which brings it to the number two position. Potential Backdoor, which is issues such as hardcoded passwords and keys, is also uniquely popular to C/C++.

The high prevalence of these top categories suggests that developers can make their applications much more secure by focusing on just a few vulnerability categories during their SDLC. C/C++ programmers need to focus on buffer issues, numeric issues, and backdoors. Web developers need to focus on XSS, CRLF Injection, SQL Injection, Command Injection, Directory Traversal, and Information Leakage. This focused approach lines up quite well with the top four hacking root cause issues listed in the Verizon 2010 Data Breach Investigations Report: Backdoor/Control Channel, SQL Injection, Command Injection, XSS.[5]

C/C++ programmers need to focus on buffer issues, numeric issues, and backdoors. Web developers need to focus on XSS, CRLF Injection, SQL Injection, Command Injection, Directory Traversal, and Information Leakage.

[5] www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

Vulnerability Distribution by Language

Java		ColdFusion		C/C++		.NET		PHP	
Cross-site Scripting (XSS)	50%	Cross-site Scripting (XSS)	89%	Buffer Overflow	27%	Cross-site Scripting (XSS)	44%	Cross-site Scripting (XSS)	80%
CRLF Injection	17%	SQL Injection	9%	Error Handling	23%	Information Leakage	23%	Directory Traversal	8%
Information Leakage	14%	OS Command Injection	<1%	Potential Backdoor	22%	Cryptographic Issues	11%	SQL Injection	6%
Cryptographic Issues	5%	Information Leakage	<1%	Numeric Orders	11%	Directory Traversal	8%	Information Leakage	3%
Directory Traversal	4%	Directory Traversal	<1%	Buffer Mgmt Errors	9%	Insufficient Input Validation	6%	Code Injection	1%

Table 4: Vulnerability Distribution by Language

Vulnerability Distribution by Analysis Type

There are three prevalent testing techniques for application security:

- Automated static analysis
- Automated dynamic analysis
- Manual testing

Veracode provides automated static analysis for all types of applications and automated dynamic and manual testing for web applications. Each testing technique has strengths and weaknesses and there is no silver bullet. In many situations, such as with high business risk applications it is prudent to use multiple testing techniques. For other situations there may only be enough resources to perform one type of testing. To enable this decision making it is useful to compare automated static and automated dynamic analysis test results to see the efficacy of the different methods in finding the vulnerability categories that are putting your organization at risk.

Table 5 lists the mean flaws detected per application by vulnerability category for all the applications that Veracode performed both static and dynamic analysis on. The applications under test are all web applications. The table is sorted by static analysis prevalence.

The first data point that jumps out is the sum of the top 15 categories is much higher for static vs. dynamic: 635 vs. 29, which is a 22 times difference. The majority of this drastic difference in quantity is caused by the top two categories: XSS and CRLF Injection which make up 77% of the static results.

If you look at the nature of XSS it is something that can be reported on every output of a web application containing untrusted user data that is not output encoded. Because static analysis can view every single output of the application, even rare edge and error outputs, static analysis gets excellent code coverage for detecting XSS. Dynamic analysis is challenged to find all the rare outputs of the application because it is impossible to create every possible state of the application in a reasonable amount of time. For XSS detection, static analysis has a great advantage over dynamic analysis.

CRLF Injection is an example of a flaw that can occur in places within an application where dynamic analysis has no visibility. Many instances of CRLF Injection occur during logging which dynamic analysis cannot see because it is internal to the application.

Static vs. Dynamic: Mean Flaws Detected per Application by Vulnerability Category

Vulnerability	Static	Dynamic
Cross-site Scripting (XSS)	354	5
CRLF Injection	133	0
Information Leakage	41	20
SQL Injection	32	4
Cryptographic Issues	18	<1
Encapsulation	16	0
Directory Traversal	14	0
Insufficient Input Validation	8	0
Race Conditions	5	0
Potential Backdoor	4	0
Time and State	4	0
Credentials Management	3	0
API Abuse	2	0
OS Command Injection	1	<1
Error Handling	<1	0

Table 5: Static vs. Dynamic: Mean Flaws Detected per Application by Vulnerability Category

Information leakage is a category where dynamic analysis does quite well compared to static analysis. Static finds more flaws but it is only a two to one ratio in number of flaws found. Both techniques were able to find SQL Injection although static analysis found about 8 times as many for this set of applications.

What accounts for the disparity between static and dynamic methods, independent of vendor? One major contributing factor is that static analysis provides comprehensive coverage of the application whereas dynamic analysis only tests code paths that it can discover externally. Often, dynamic (and even manual) testing completely overlooks portions of the application that are only reachable under certain circumstances. For example, application functionality may be gated behind a series of forms that trigger different behavior depending on how they are filled out. Also, applications that support different types of users (e.g. view-only, author, editor, administrator, power user, etc.) often restrict the functionality that each user level can access, meaning that the application must be scanned multiple times, iterating over all of the user roles, in order to maximize coverage.

The lesson for CISOs and CIOs is that a robust application security program must incorporate multiple testing methods in order to ensure that applications are assessed with sufficient coverage, measured by both depth and breadth. Becoming overly dependent on too few analysis methodologies guarantees blind spots when assessing overall application risk.

The lesson for CISOs and CIOs is that a robust application security program must incorporate multiple testing methods in order to ensure that applications are assessed with sufficient coverage, measured by both depth and breadth. Becoming overly dependent on too few analysis methodologies guarantees blind spots when assessing overall application risk.

Distribution of Vulnerabilities by Industry

Industries experience differing levels of cyber threats, may employ web or non-web applications of differing criticality, use programming languages to different degrees, and have vary in maturity with respect to application risk management. As a result, comparisons across very different industries can be challenging, although comparisons within industry sub-segments (see Software Industry Analysis section). Table 6 contains the vulnerability distribution by prevalence within an industry vertical.

It is not a surprise that XSS is the first or second most prevalent risk for all the categories but Government applications stands out as being the worst on XSS issues. On the other hand Government applications do the best on Information Leakage out of all the industries.

The only other category that is a standout by industry is prevalence of potential backdoor flaws found for the Software industry. This is typically hardcoded passwords and keys and is often just poorly implemented debug or support functionality. It makes sense that the Software industry, which often provides remote product support for customer deployments, would have a greater prevalence of this vulnerability.

Vulnerability Distribution by Industry

Finance		Software		Government		Other	
Cross-site Scripting (XSS)	58%	Information Leakage	22%	Cross-site Scripting (XSS)	81%	Cross-site Scripting (XSS)	48%
Information Leakage	11%	Cross-site Scripting (XSS)	17%	SQL Injection	7%	CRLF Injection	16%
CRLF Injection	11%	CRLF Injection	9%	CRLF Injection	4%	Information Leakage	14%
Cryptographic Issues	4%	Cryptographic Issues	8%	Cryptographic Issues	2%	Cryptographic Issues	4%
Encapsulation	3%	Directory Traversal	8%	Information Leakage	2%	SQL Injection	4%
SQL Injection	3%	Potential Backdoor	7%	Directory Traversal	<1%	Directory Traversal	4%
Directory Traversal	3%	Buffer Overflow	6%	Time and State	<1%	Encapsulation	2%
Insufficient Input Validation	1%	Error Handling	6%	Insufficient Input Validation	<1%	Time and State	2%
Buffer Overflow	1%	Numeric Errors	3%	OS Command Injection	<1%	Buffer Overflow	1%
Time and State	1%	Time and State	3%	Credentials Mgmt	<1%	Potential Backdoor	<1%
Error Handling	<1%	SQL Injection	3%	Encapsulation	<1%	Error Handling	<1%
Potential Backdoor	<1%	Buffer Mgmt Errors	2%	Error Handling	<1%	Untrusted Search Path	<1%
Credentials Mgmt	<1%	Credentials Mgmt	1%	Potential Backdoor	<1%	Insufficient Input Validation	<1%
Race Conditions	<1%	Encapsulation	<1%	API Abuse	<1%	Credentials Mgmt	<1%
API Abuse	<1%	API Abuse	<1%	Buffer Mgmt Errors	<1%	API Abuse	<1%

Table 6: Vulnerability Distribution by Industry

Industry Group Definitions: *The Finance-related industries group combines applications from the Financial Service, Insurance, and Banking industries (self identified); the Computer-related industries category combines applications from the Computer Software, Computer Services, and Security Products and Services industries; Government is unclassified US federal, state, and local government agencies (self-identified). Other combines applications from from all other industries including Energy, Healthcare, Pharmaceuticals, Media and Entertainment, Computer Hardware, Manufacturing, Education, Aerospace and Defense and Telecommunications (self identified).*

Distribution of Application Security Performance by Business Criticality

Veracode's risk adjusted benchmark applies a sliding scale, requiring applications of higher business criticality (assurance levels) to have a higher degree of security quality (refer to addendum for detailed methodology). This pragmatic approach allows organizations to optimize their remediation effort and make judicious use of their application security funds.

We investigated the performance of applications upon initial submissions relative to their business criticality. As Figure 23 shows only 14% of applications designated as "Very High" business criticality were deemed to have acceptable performance on initial submission. This is a significant downturn from the last report where 32% of the applications were found to be of acceptable security quality. However, results from applications of "High" business criticality remained the same with 38% acceptable upon initial submission. Performance on initial submission of the most mission-critical applications in an organization remains poor. Clearly a lot needs to be done to improve the security posture of the most mission critical applications in an organization.

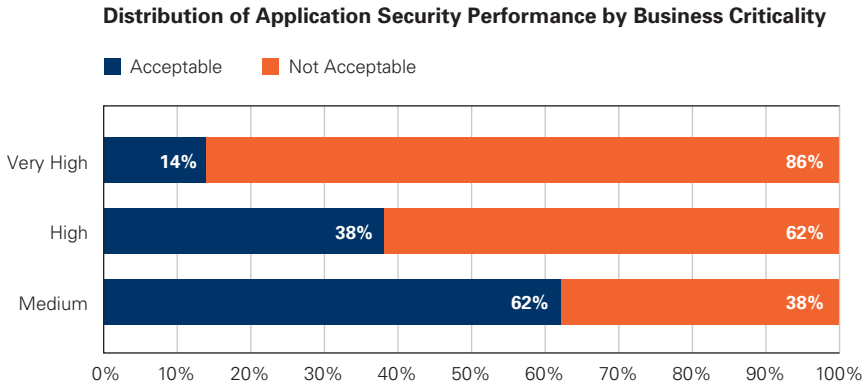


Figure 23: Distribution of Application Security Performance by Business Criticality

We explored the impact of industry on the application performance above, to determine whether some industry verticals have more security-aware development teams and more formal development practices. The results of our analysis in Figure 24 show Government and Financial continue to maintain the top two spots. Software-related Industries continue to fare the worst with only 34% found to be acceptable. Given that this has been a consistent theme in our reporting we decided to perform a deep dive of the Software industry to help understand why this may be the case.

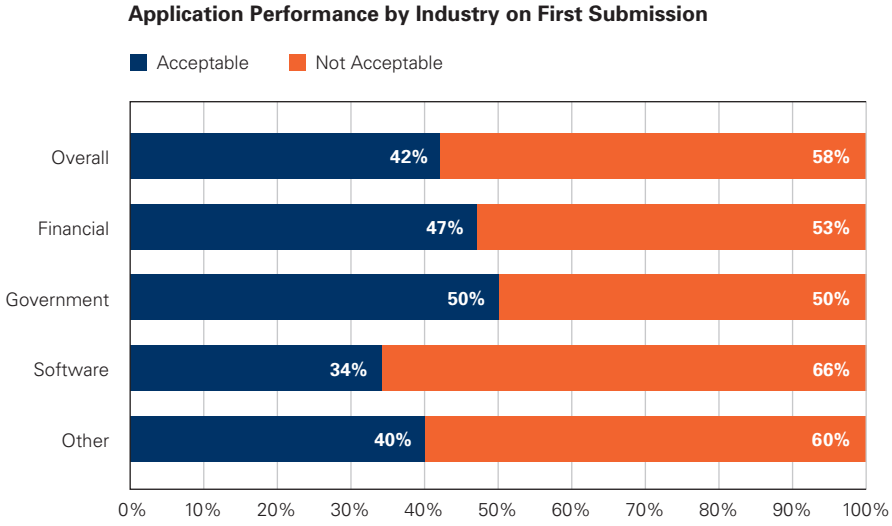


Figure 24: Application Performance by Industry on First Submission

Software Industry Analysis

Readers may recall that in Volume 2 we explored the Financial Industry in more depth and compared Financial Services to Insurance to Banks. In this report we dive deeper into the Software industry with the objective of understanding relative performance and trends exhibited by different sectors within this industry group. The Software industry category comprises at a high-level computer software, computer services, and security products and services. We have further dissected this category along several axes such as whether the company is privately held or not, revenue run rate etc. It was our goal to understand whether these aspects of a software company have any bearing on the security posture of the applications it is producing.

Security Quality Score Distribution by Company Type

Figure 25 represents the raw security quality scores upon initial submission obtained upon by privately held and publicly traded software companies. We explored whether the public nature of a company with the additional scrutiny and regulatory oversight would have any bearing on the security posture of their applications. It appears not to be the case though. As Figure 25 shows the distribution as well as the median security quality score of applications from both private and public companies are very closely aligned. Both privately held and publicly traded companies had a median security quality score between 70 and 80. This can be instructive to software purchasers who sometimes may favor a public company over a private one with similar functioning products due to an assumed posture of greater security.

Surprising to see no discernable difference in the raw security quality score upon initial submission of applications from private vs. public software companies.

Security Quality Score Distribution for Public vs. Private Software Company

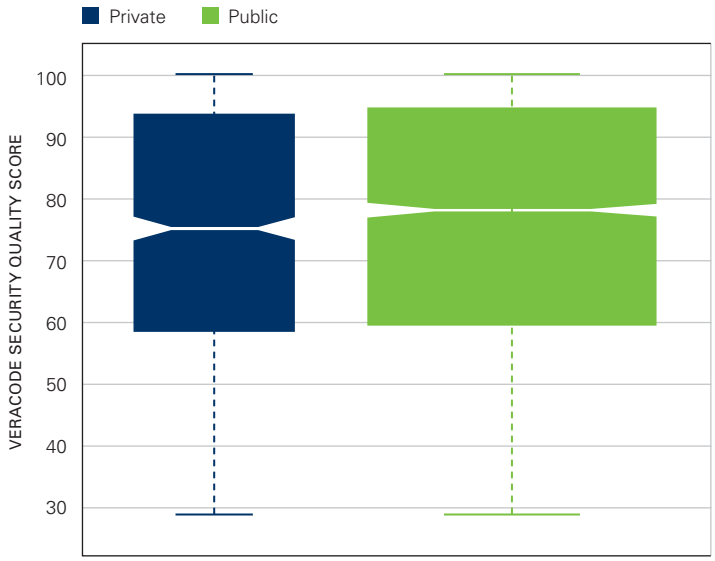


Figure 25: Security Quality Score Distribution for Public vs. Private Software Company

Security Quality Score Distribution by Software Company Revenue

Next we examined whether the overall revenue of the company had any bearing on the security quality of the applications being produced as measured by the security quality score upon initial submission. As Figure 26 demonstrates, both the median security scores and the score distributions are pretty closely aligned. The median security quality score was between 70 and 80 across companies of all revenue.

Security Quality Scores upon initial submission are similar for companies across all revenue brackets.

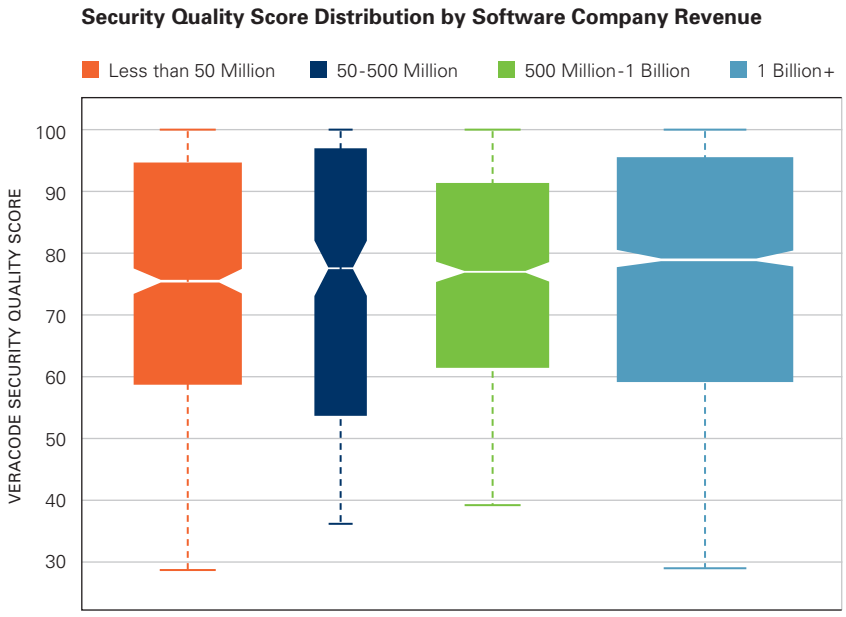


Figure 26: Security Quality Score Distribution by Software Company Revenue

Security Quality Score Distribution by Application Purpose

Figure 27 represents the raw security quality scores upon initial submission obtained by software companies based on the type of software they are producing. It should be noted that very large companies that have multiple software products and services were categorized on the basis of the application purpose for the majority of their products. Not surprisingly software companies producing financial software ranked the best in terms of the raw security quality score. This is a consistent theme in our reporting where applications or organizations relating to financial data and processes exhibit a higher level of security performance than other industries or application purposes. This may be due to greater regulation and oversight of these applications and organizations or simply because the connection between a security weakness and monetary loss is more readily apparent.

Customer Support and Content Management and Collaboration Software were the next two highest scoring categories with median scores of 74 and 78 respectively. Surprisingly Operations and Security Products and Services were at the bottom of the stack with median scores of 73 and 76 respectively. This was disappointing given the criticality of these types of software applications to the reliable and secure functioning of an organization's technology infrastructure.

Security Quality Score Distribution by Application Purpose

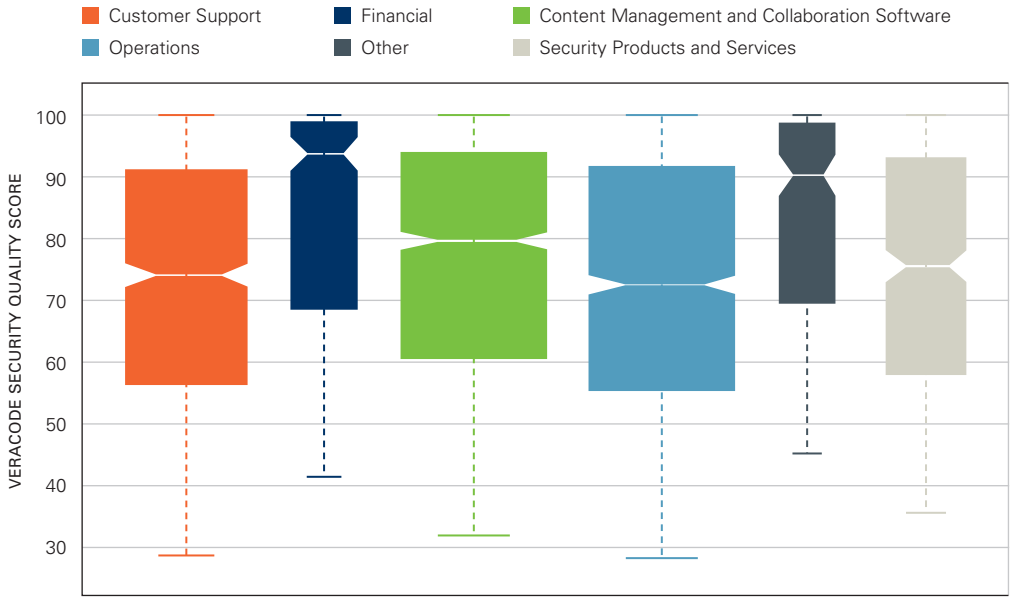


Figure 27: Security Quality Score Distribution by Application Purpose

Application Purpose Definitions: *Customer Support category includes customer relationship management, web customer support and other types of customer support applications; Financial category includes traditional accounting and finance applications and newer mobile banking applications; Content Management and Collaboration Software category includes website content management, content collaboration and sharing and web conferencing types of applications; Operations category includes applications supporting day-to-day non-financial business activity such as product development, information management utilities, IT management and SDLC tools etc.; Security Products and Services category includes applications that are being used to perform a security function; Other is a combination of remaining types of applications and includes healthcare applications.*

Application Security Performance by Software Sub-segment

Next we factored in the business criticality of the applications to see what the acceptable rating would be against the Veracode risk adjusted verification methodology. Overall, 66% of applications from the Software industry were found to be unacceptable upon initial submission. This is higher than the average of 58% for all applications included in this reporting period (depicted in Figure 28). Security Products and Services and Customer Support categories fared the worst within the Software industry segment. This is presumably due to the higher business criticality associated with these applications.

Security Products and Services and Customer Support categories fared the worst within the Software industry segment.

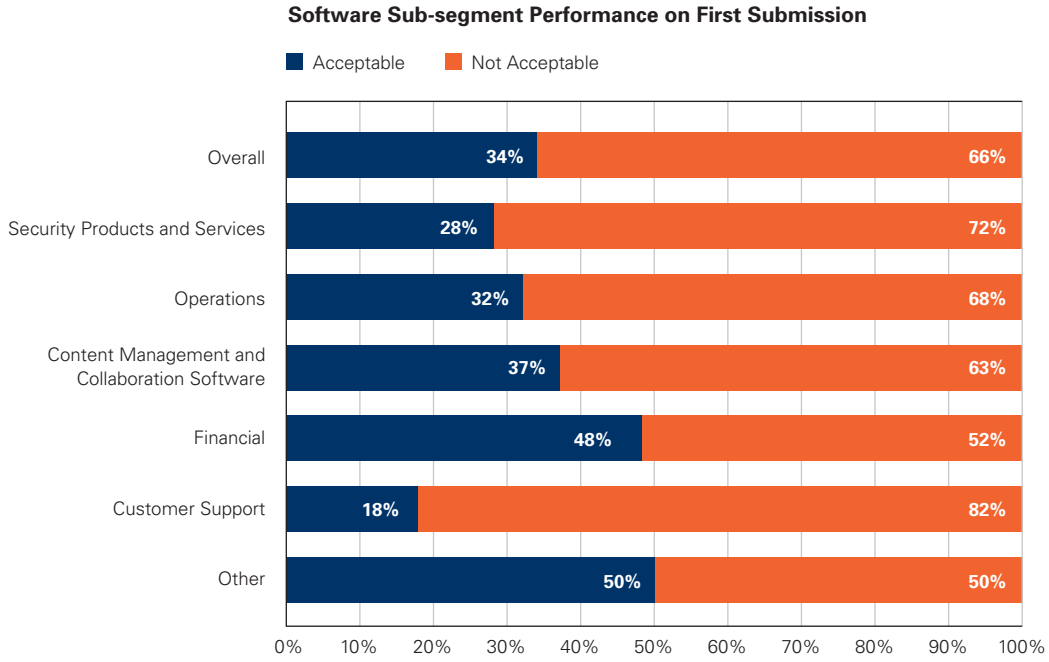


Figure 28: Software Sub-segment Performance on First Submission

Time to Acceptable Security Quality by Software Sub-segment

The earlier figures shed light on the raw security quality scores upon initial submission and the acceptance rate once you layer on the notion of business criticality of those applications. What becomes increasingly evident is that much work needs to be done by the Software industry to improve the security posture of its applications. So, let's examine how quickly this work is being performed by taking a look at the remediation behavior exhibited by this industry segment in Figure 29. Over 90% of applications across the different sub-sectors of the Software industry achieve the acceptable security quality within 1 month (as measured against Veracode's risk adjusted verification methodology). In the case of Security Products and Services 98% achieved the requisite policy within 1 month. In fact, the average time for the Security Products and Services was 3 days. So, while the initial security quality score for some applications may have been poor the overall time to achieve acceptable security quality is swift.

More than 90% of applications across Software industry sub-sectors achieve the acceptable security quality within 1 month. In the case of security products and services 98% achieved the acceptable security quality within 1 month.

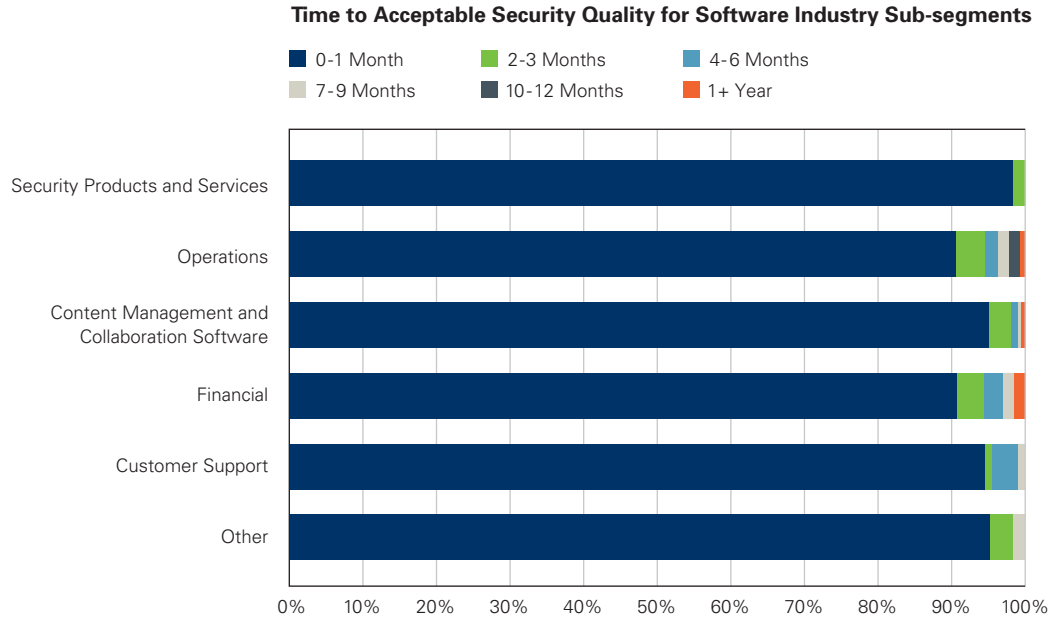


Figure 29: Time to Acceptable Security Quality for Software Industry Sub-segments

Figure 30 shows the percentage distribution by sub-segment for the dataset used for the Software industry analysis.

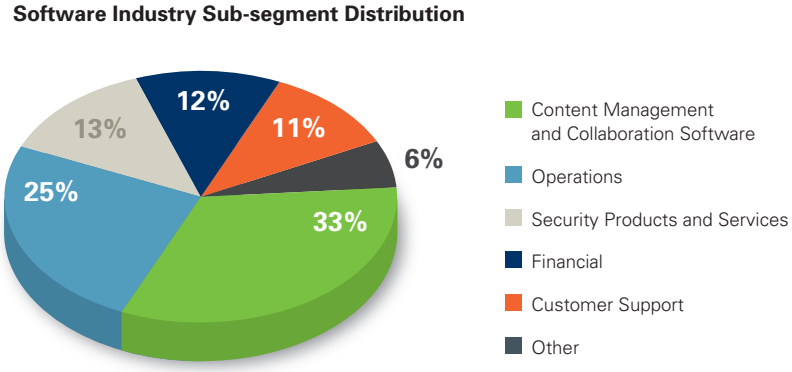


Figure 30: Software Industry Sub-segment Distribution

Developer Training and Education

Developer education is widely considered to be a key part of an application security program. As Veracode provides a cloud based eLearning capability as part of its service, this report looks at some baseline information on the performance of developers against several graded assessments to gain an understanding of how the skill level of developers may or may not contribute to application security risk (Figure 31).

The Veracode eLearning service offers several graded exams that are either taken by themselves or following an eLearning course. This report looks at the performance of developers on four assessments: the Veracode Application Security Fundamentals Assessment, the Secure .NET Coding exam, the Secure Java Coding exam, and the Introduction to Cryptography exam. For the purposes of this analysis, a grading scale was assigned to the results: A: >90; B: between 80 and 89; C: between 70 and 79; D: between 60 and 69; F: less than 60.

More than 50% of students achieved a rating of C or lower when tested for application security fundamentals. More than 30% had a failing grade of D or F.

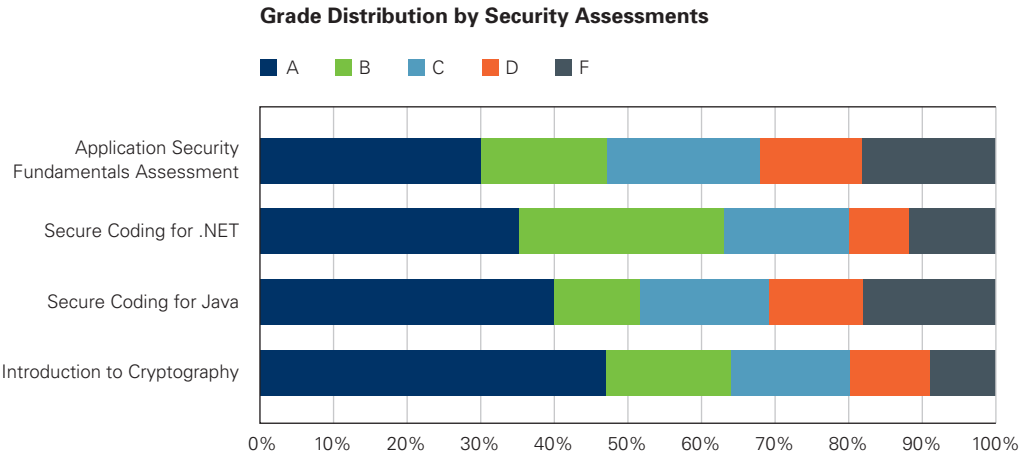


Figure 31: Grade Distribution by Security Assessments

Measurements of initial security knowledge: While a customer is free to take the assessments in any order, the Veracode Application Security Fundamentals Assessment is commonly taken as the first step in the eLearning program as a skills assessment to determine future coursework. The assessment covers knowledge of security concepts, including common threats, and may be taken by developers, managers, or QA testers. This assessment had the lowest number of students achieving an A (29%) and the highest number achieving a D or F (31% combined). This suggests that while there is a sizable population of developers who have a good knowledge of security fundamentals, there is also a large population who have a poor grasp of the fundamentals of application security. It should also be noted that this test is generally taken before any coursework to evaluate the baseline understanding of users on common application security principles. That may also explain lower scores as compared to the other tests which are generally taken after appropriate coursework.

Measurements of developer knowledge: The two Secure Coding exams are targeted at developers, and are generally taken following structured courses on their topics and include questions that ask the student to evaluate the security of code samples as well as concept testing. While a larger portion of users scored an A on these tests (35% for .NET, 40% for Java), there were still a substantial number who scored a D or F. Of note, the same percentage (31%) of the students taking the Java exam scored a D or F as on the Security Fundamentals Assessment.

While 40% of students scored an A on Secure Coding for Java, more than 30% got failing grade of D or F. 35% of students scored an A on Secure Coding for .NET while 20% got failing grade of D or F.

Topic specific knowledge (cryptography): Finally, a larger portion of students scored an A on the Introduction to Cryptography course than on any other exam (47%), but there were still 20% of the students who scored a D or F. Apparently, while many developers understand the concepts and security implications of cryptography, a significant minority do not, and this may provide some context to explain the relatively high position of cryptographic flaws in the flaw prevalence charts above.

Figure 32 represents the percentage of students taking the different courses depicted in Figure 31.

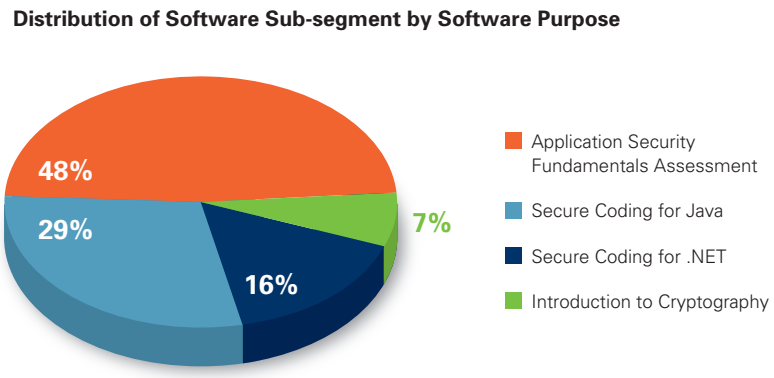


Figure 32: Distribution of Software Sub-segment by Software Purpose

Application Threat Space Trends

The state of software security in 2011 hasn't changed much in the last 6 months but the threat space continues to worsen. Application vulnerabilities are being used by attackers to penetrate into corporations to steal intellectual property. In the beginning of 2011, SQL Injection was used in high profile attacks such as those on the security company HBGary, the Night Dragon attacks on energy companies, and the certificate forging attack on Comodo. SQL Injection vulnerabilities in applications on organization perimeters were leveraged to get insider access to corporate secrets. Another major attack of early 2011 used a software flaw in a desktop application, Adobe Flash, to bridge the perimeter and install remote access software on an unwitting employee's desktop as a way of penetrating further into RSA and stealing valuable corporate secrets.

The above corporate breaches are examples of the two major threat space trends going on today.

1. Attackers are discovering and exploiting common vulnerabilities on internally developed web applications to steal the data they manage or as stepping stones to penetrate deeper into organizations.

2. Attackers are taking advantage of vulnerabilities found in common desktop software purchased by organizations in order to compromise employee workstations as pivot points to get deeper toward their goal.

These trends will continue until solutions are put in place to diminish the quantity of these software vulnerabilities or mitigate their exploitability. There is just too much at stake. Traditional perimeter defense and detection software has been shown to be woefully inadequate against these major threat space trends. We don't need more security software. We need more secure software.

Traditional perimeter defense and detection software has been shown to be woefully inadequate against these major threat space trends. We don't need more security software. We need more secure software.

Some organizations are stepping up to these enhanced attack trends and putting in place comprehensive application security or application risk management programs. These programs seek to instill application security across the entire organization application portfolio, whether the software is developed in-house or delivered from an external source such as an outsourced, open source project or traditional software vendor. Most organizations that have a relatively mature internal SDLC process are moving to add a third-party risk or supply chain risk program.

The majority of corporations and government organizations are not at this level of maturity and are still tackling application security as a project covering just their highest risk internal applications. This isn't nearly enough. HBGary was compromised with a SQL Injection vulnerability in a modified open source CMS. Night Dragon attackers hit any vulnerable perimeter web application at the energy companies they targeted. RSA was done in by a vulnerability in a common software component for document viewing. It is safe to say that the vulnerable software that allowed these organizations to be penetrated was not covered by their application security projects.

Besides the threat space trends wreaking new havoc on the traditional desktop and server platforms there are new non-traditional platforms coming on line in the corporate and government world: mobile and cloud. Cloud platforms came first, but mobile is growing like wildfire and is becoming part of the enterprise ecosystem faster than a CEO can say, "Hook my iPad up to the Exchange server." It isn't just executives though. Everyone wants to do more enterprise computing on the new mobile platforms.

RIM's Blackberry platform has been a mainstay in corporate and government IT departments for many years. The platform was built for enterprise worthy email and device security. Perhaps in an odd way its security was enhanced by lack of many 3rd party apps. New mobile platforms including iOS and Android are changing all this with a lack of enterprise features and a rich consumer ecosystem of third-party apps. While there are many thousands of compelling apps for these new mobile platforms and that is a great part of their appeal, all it takes is a few dangerous ones to leave enterprise security teams scrambling for solutions to keep their mobile users productive and happy while protecting corporate assets from exposure.

The way to secure these new mobile platforms for the enterprise is still evolving but it is clear something has to happen at the app layer. The Android marketplace has allowed a few trojaned applications to be downloaded by users with the DroidDream attack being the largest, downloaded by over 50,000 users. The walled garden approach of Apple has shown itself to be more secure due to the higher scrutiny of the approval process. This ecosystem is incredibly dynamic so mobile app security, in its infancy now, will likely be in for tremendous growth in 2011.

Organizations will continue to run more applications in cloud environments, but unlike mobile, enterprises are moving more cautiously with this new environment. Efforts such as the Cloud Security Alliance are helping to educate enterprise development teams and security departments how to build secure applications for the cloud.

Because of the threat space trends and new platforms, applications security continues to be a very dynamic arena. Organizations will need to be nimble and scale up to many applications over many different platforms to keep pace with attackers in 2011.

Because of the threat space trends and new platforms, applications security continues to be a very dynamic arena. Organizations will need to be nimble and scale up to many applications over many different platforms to keep pace with attackers in 2011.

Addendum

Methodology

About Veracode's Risk Adjusted Verification Methodology

The Veracode SecurityReview uses static and dynamic analysis (for web applications) to inspect executables and identify security vulnerabilities in applications. Using both static and dynamic analysis helps reduce false negatives and detect a broader range of security vulnerabilities. The static binary analysis engine creates a model of the data and control flow of the binary executable; the model is then verified for security vulnerabilities using a set of automated security scans. Dynamic analysis uses an automated web scanning technique to detect security vulnerabilities in a web application at runtime. Once the automated process is complete, a security analyst verifies the output to ensure the lowest false positive rates in the industry. The end result is an accurate list of security vulnerabilities for the classes of automated scans applied to the application.

About Software Assurance Levels

The foundation of the Veracode rating system is the concept that higher assurance applications require higher security quality scores to be acceptable risks. Lower assurance applications can tolerate lower security quality. The assurance level is dictated by the typical deployed environment and the value of data used by the application. Factors that determine assurance level include reputation damage, financial loss, operational risk, sensitive information disclosure, personal safety, and legal violations.

About the Data Set

The data represents 4,835 applications submitted for analysis by large and small companies, commercial software providers, open source projects, and software outsourcers. An application was counted only once even if it was submitted multiple times as vulnerabilities were remediated and new versions uploaded. The report contains findings about applications that were subjected to static, dynamic, or manual analysis through the Veracode cloud Platform. The report considers data that was provided by Veracode's customers (application portfolio information such as assurance level, industry, application origin) and information that was calculated or derived in the course of Veracode's analysis (application size, application compiler and platform, types of vulnerabilities, Veracode rating).

In any study of this size there is a risk that sampling issues will arise because of the nature of the way the data was collected. For instance, it should be kept in mind that all the applications in this study came from organizations that were motivated enough about application security to engage Veracode for an independent assessment of software risk. Care has been taken to only present comparisons where a statistically significant sample size was present.

About the Findings

Unless otherwise stated, all comparisons are made on the basis of the count of unique application builds submitted and rated.

Assurance Level Definitions

Veracode's Business Criticality designations are based on the Assurance Level standard developed by NIST, as detailed below:

Business Criticality	Description
Very High	Mission critical for business/safety of life and limb on the line
High	Exploitation causes serious brand damage and financial loss with long term business impact
Medium	Applications connected to the Internet that process financial or private customer information
Low	Typically internal applications with non-critical business impact
Very Low	Applications with no material business impact

Table 7: Business Criticality Descriptions
Source: U.S. Government. OMB Memorandum M-04-04; NIST FIPS Pub. 199

Very High (AL5)

This is typically an application where the safety of life or limb is dependent on the system; it is mission critical the application maintain 100% availability for the long term viability of the project or business. Examples are control software for industrial, transportation or medical equipment or critical business systems such as financial trading systems.

High (AL4)

This is typically an important multi-user business application reachable from the Internet and is critical that the application maintain high availability to accomplish its mission. Exploitation of high assurance applications cause serious brand damage and business/financial loss and could lead to long term business impact. Exploitation is a result of a breach in any two impact categories of confidentiality, integrity and availability of the application.

Medium (AL3)

This is typically a multi-user application connected to the Internet or any system that processes financial or private customer information. Exploitation of medium assurance applications typically result in material business impact resulting in some financial loss, brand damage or business liability. Exploitation is a result of a breach in confidentiality, integrity or availability of the application. An example is a financial services company's internal 401K management system.

Low (AL2)

This is typically an internal only application that requires low levels of application security such as authentication to protect access to non-critical business information and prevent IT disruptions. Exploitation of low assurance applications may lead to minor levels of inconvenience, distress or IT disruption. An example internal system is a conference room reservation or business card order system.

Very Low (AL1)

Applications that have no material business impact should its confidentiality, data integrity and availability be affected. Code security analysis is not required for this assurance level and security spending should be directed to other higher level assurance applications.

VERACODE

SSSR/0411

ABOUT VERACODE

Veracode is the only independent provider of cloud-based application intelligence and security verification services. The Veracode platform provides the fastest, most comprehensive solution to improve the security of internally developed, purchased or outsourced software applications and third-party components. By combining patented static, dynamic and manual testing, extensive eLearning capabilities, and advanced application analytics Veracode enables scalable, policy-driven application risk management programs. Veracode delivers unbiased proof of application security to stakeholders across the software supply chain while supporting independent audit and compliance requirements for all applications no matter how they are deployed, via the web, mobile or in the cloud. The company's more than 175 customers include Barclays PLC, California Public Employees' Retirement System (CalPERS), Computershare and the Federal Aviation Administration (FAA). For more information, visit www.veracode.com, follow on Twitter: @Veracode or read the ZeroDay Labs blog.