VOLUME 4

State of Software Security Report

The Intractable Problem of Insecure Software

December 7, 2011

Now Including
Mobile App Data!
SEE PAGE 37

VERACODE

SINCE OUR LAST REPORT, the risks associated with vulnerable software deployed in enterprise environments have been highlighted in the news on nearly a weekly basis. The majority of reported breaches that exposed customer data or intellectual property were caused by attackers exploiting weaknesses in web applications or desktop software. We have also witnessed the rise of new attacker categories: cyber spies focused on stealing intellectual property, and hacktivists motivated by publicly embarrassing companies and individuals. Essentially, if your organization has anything worth protecting—money, intellectual property or a trusted reputation—you need to be concerned about the security of the software that is embedded within every aspect of your enterprise.

Over the horizon, we see changes that are adding risk, not reducing it. The risks associated with the consumerization of IT have gripped enterprise security teams as they face unrelenting threats, often at the hands of their very own employees. They lack consistent, proactive policies to manage vulnerabilities associated with the "BYOD" or Bring Your Own Device trend. In fact, according to a Wall St. Journal article (Sept. 26, 2011), "Danger-to-Go," mobile devices "come with one huge challenge: How do you make sure all that valuable information is secure?" Further, "the increasing presence of such devices elevates the threat of accidental and intentional security breaches." Based on what we're witnessing among our customer base, verifying the security of the software being downloaded to those devices is increasingly becoming a business priority. This is especially true when it is believed that platforms like Google's Android do minimal vetting of the safety of applications they allow consumers to download from their App store.

In version 4 of our State of Software Security report, we continue our analysis and examine emerging trends associated with vulnerabilities in applications—whether they are internally developed or procured from third-parties such as outsourcers or commercial software vendors. For the first time, we also take a closer look at Android security trends and highlight key takeaways for organizations seeking to balance employee mobility and productivity against mobile security risk.

With multiple pieces of proposed legislation currently being examined by Congress, the topic of federal cyber security has garnered attention in recent months, suggesting that it is an area warranting further investigation in this report. With renewed commitments to sharing data breach information and evolving best practices for more effectively managing cyber security threats, we have witnessed several positive steps being taken across the public and private sectors.

That message was emphasized last summer when I had the opportunity to meet with Senate staffers on the Homeland Security and Governmental Affairs Committee in order to help create better awareness and understanding about cyber attacks against the Government and private firms. As part of that experience, we realized the importance of educating federal audiences about cyber security risks in a vendor agnostic role. To that end, in this report we delved deeper into the nature of government software applications and how they fare relative to peers in private industry sectors.

In this volume, Veracode analyzed more than 9,000 application builds, across 40 different industry sectors. In addition to mobile application and government security trends analysis, we revisit several findings highlighted in previous reports. This includes measuring the security quality of third-party software from large and small software vendors, at the request of enterprises, continuing to debunk the myth that any vendor is "too big to test." We also reexamine the impact of dedicated training and ongoing educational programs for software developers and security managers while updating our list of the most common software vulnerabilities that put entire software portfolios at risk.

As you examine the information presented here, we welcome your questions and ideas about what we can do as an industry to ensure a long-term commitment to protecting our software infrastructure and continuing to raise the visibility of software-related business risk.

Best Regards,

Chris Wysopal
Founder, CISO and CTO, Veracode

Table of Contents

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 representing thousands of applications.

The resulting security intelligence is unique in the breadth and depth it offers. It represents multiple testing methodologies (static binary, dynamic, and manual) on the full spectrum of application types (components, shared libraries, web, non-web applications, mobile applications) and programming languages (including Java, C/C++, .NET languages, ColdFusion, PHP, and Objective C) from every part of the software supply chain (Internally Developed, Open Source, Outsourced, Commercial). For executives, security practitioners and developers who want to understand the root cause of recent breaches, this is essential reading.

This volume captures data collected over the past 18 months from the analysis of 9,910 application builds on our cloud platform (compared to 4,835 application builds in Volume 3 published in April 2011). 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 (including mobile).

New in Volume 4 are sections on Android application analysis, vulnerability prevalence as viewed from the lens of the threat space (i.e. most frequently exploited vulnerabilities) and a comparative analysis of the Government sector relative to other industries and the overall dataset.

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

Greater than two-fold increase in number of application builds included in Volume 4 (9,910) compared to Volume 3 (4,835) indicating increased activity on part of customers and growing use of independent cloud-based application security services.

Executive Summary

The following are some of the most significant findings in the Veracode State of Software Security Report, Volume 4, representing 9,910 application builds assessed in the last 18 months by Veracode on our cloud-based application security platform.

1. Application security performance declines steeply when current threat landscape is taken into account in the evaluation criteria
2. Vulnerabilities that can lead to remote code execution and backdoor functionality are found to be far more prevalent in commercial software
3. Cross-site Scripting and SQL Injection were found to affect higher percentage of Government applications than other industry sectors; SQL Injection trend is flat while declining in the overall dataset
4. A sizable proportion of Android applications were found to contain hard-coded cryptographic keys
5. Independent security verification of third-party software is being carried out by multiple industry segments
6. Greater knowledge of application security is associated with improved security quality scores
7. Development agility and application security are not mutually exclusive

Key Findings

1. Application security performance declines steeply when current threat landscape is taken into account in the evaluation criteria

Earlier this year Veracode transitioned from its risk-adjusted verification methodology used in prior volumes to a new policy intended to be more resilient against prevalent threats. This new policy makes the acceptability threshold more stringent, particularly for higher criticality applications. For example, it adopts a zero-tolerance policy towards frequently exploited vulnerabilities such as Cross-site Scripting (XSS) and SQL Injection (root cause of the Sony breach and countless others). Even a single instance of these types of vulnerabilities causes applications of higher criticalities to be deemed unacceptable. The result of this new policy on application performance was drastic. Over 8 in 10 applications across all supplier types failed to pass when first tested. This marks a steep decline from Volume 3 where the policy being used to measure acceptability was a bit more forgiving and in comparison only 58% of applications failed upon initial submission.

Recommendation: Vulnerabilities such as Cross-site Scripting (XSS) and SQL Injection continue to make headlines because they are frequently exploited by attackers. According to the Web Hacking Incident Database, 20% of reported incidents were caused by SQL Injection. This stands to reason as they are highly prevalent (as Figure 20 shows XSS is present in 68% of all web applications and SQL Injection is present in 32% of all web applications) so they present the "low hanging fruit." Given this threat environment organizations should implement a program that allows for the discovery and timely remediation of such vulnerability types. Leveraging automated testing technologies allows organizations to scale such a program to their entire application inventory. Furthermore, organizations should adopt a policy where any occurrence of these types of frequently exploited vulnerabilities is not tolerated.

2. Vulnerabilities that can lead to remote code execution and backdoor functionality are found to be far more prevalent in commercial software

Internally developed, outsourced, and open source software development continues to shift to managed code languages and scripting languages which don't fall victim to many of the more serious vulnerability categories that can lead to remote code execution. These vulnerabilities, including buffer management errors, buffer overflows, and integer overflows are found much more often in commercial software due to the continued use of compiled C/C++ and Objective C languages.

An additional high risk vulnerability category, backdoor functionality, is also more prevalent in commercial software. Backdoor functionality, whether malicious or by design, is a category that all buyers of commercial software should be mindful of.

Recommendation: As we have demonstrated since Volume 1 of this report, there is meaningful reliance across industry sectors on commercial suppliers for business critical applications. This is causing organizations to seek independent security verification of commercial software as part of their procurement process (Refer to executive summary finding 5 for more details on third-party risk assessments). Understanding the unique nature of vulnerabilities afflicting commercial software can help inform the policy that should be applied to these applications for them to be considered acceptable. Remote code execution vulnerabilities and backdoor functionality should be explicitly scanned for when testing these applications. A clear requirement for C/C++ and Objective C language support also emerges when choosing security testing technologies for third-party software.

3. Cross-site Scripting and SQL Injection were found to affect higher percentage of Government applications than other industry sectors; SQL Injection trend is flat while declining in the overall dataset

We carried out a deeper analysis of U.S. Government applications (representing federal, state and local government) to investigate whether there are any key similarities or differences between those applications and the remainder of the dataset. Many of these applications process critical data such as PII, national security data and operate critical systems. What we found was that Government web applications had a much higher incidence of XSS issues as compared to Finance and Software—75% of Government web applications had XSS issues compared to 67% for Finance and 55% for Software (Table 7). A partial explanation of this maybe offered by the fact that Government applications utilized a higher percentage of ColdFusion than other industry segments. As Table 4 demonstrates ColdFusion has a higher incidence of XSS issues as compared to other platforms. ColdFusion also tends to be used by less experienced developers for creating web applications with greater ease. These developers are also less likely to be experienced in secure coding practices. Similarly, we found that 40% of government web applications had SQL Injection issues as compared to 29% for Finance and 30% for Software (Table 7). It was also interesting to observe that while SQL Injection was trending lower for the overall dataset (Figure 23) in Government applications it is remaining flat (Figure 30).

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 Government software applications if they are to be more resilient to cyber attacks. Government organizations are encouraged to double down on their efforts to train their development and security staff on how to avoid these errors or fix them quickly once they are found. In addition to training, diligent use should be made of automated testing techniques to expediently discover these vulnerabilities across all of their applications and to verify that the development team is following the guidance learned from their training.

4. A sizable proportion of Android applications were found to contain hard-coded cryptographic keys

This report included Android applications for the first time. Though the dataset was small—just under 1% of our dataset—we found that mobile developers tend to make the same mistakes as enterprise developers. One example of this was the practice of hard-coding cryptographic keys directly into the application. The problem is, once these keys are compromised, any security mechanisms that depend on the secrecy of the keys are then rendered ineffective. In our dataset, over 40% of Android applications contain at least one instance of this flaw (Table 5), a higher rate than we observe across all non-Android Java applications (only 17% of all non-Android Java applications had at least one instance of hard-coded cryptographic keys). More importantly, Android applications are easy to decompile, making it trivial for an attacker to extract and publicize hard-coded keys.

Recommendation: Hold mobile applications to the same security standards as other enterprise applications. Mobile applications are inherently more exposed than web applications because a motivated attacker can start reverse engineering simply by copying the executable off their phone. With that in mind, information embedded into the application—including cryptographic keys—should never be considered secret.

5. Independent security verification of third-party software is being carried out by multiple industry segments

Beginning in Volume 2, we started reporting on the third-party risk assessment market. We define this market as the independent security verification that is performed by software purchasers on applications they procure from their extended software supply chain. As the reliance on third-party software and components has grown, so has the awareness that security weaknesses embedded in those applications become a liability for the enterprise that is accepting that software. This recognition transcends the security community as you see calls for this level of due diligence from leaders in the sourcing and vendor management area as well. In this report we examined which industry segments are heeding this call to action and engaging in this process with their third-party software suppliers. We found enterprises representing at least eight different industry segments—Software, Finance, Aerospace & Defense, Government, Entertainment, Telecommunications, Insurance, and Oil & Gas. While Software and Finance account for the majority of the dataset, companies across the spectrum are starting to hold their software suppliers accountable.

Recommendation: The distribution of applications in an enterprise portfolio has remained relatively unchanged in the last few volumes of the report—approximately one-third of applications are characterized as third-party and two-thirds as internally developed. Oftentimes code developed by an outsourcing partner is labeled internally developed so the actual percentage of third-party software is believed to be even higher than reported. What has also remained true is that 30 to 70% of code components in internally developed applications are in fact third-party components and libraries. With such heavy reliance on code coming from outside an organization, a formal third-party risk assessment program becomes crucial to managing overall application risk. We recommend that all enterprises institute a policy that requires third-party vendors to demonstrate proof of independent security verification or to submit to that due diligence. We also recommend that sourcing and vendor management professionals include specific language in contracts that clarify the security threshold deemed acceptable by the enterprise and make that a requirement for any commercial transaction to proceed.

6. Greater knowledge of application security is associated with improved security quality scores

Beginning in Volume 3 we started our exploration of the state of application security knowledge amongst developers and security professionals. We reported some discouraging statistics that continue to hold true in this volume. More than half of all developers get a grade of C or lower on the application security fundamentals assessments. Top performers (those achieving grade of A) declined from 31% in Volume 3 to 27% in Volume 4 (Figure 34). However, there are some encouraging signs regarding the impact that greater proficiency in application security could have on the resultant security quality of applications being authored by those developers. In this volume we plotted the average security quality score against the average performance on the Veracode Application Security Fundamentals assessment (Figure 36). What we saw was a clustering of data points in the top right i.e. a high score on the assessment and a high score on the application scan. It would appear that those that had a high level of application security knowledge also delivered higher security quality applications. This is good news as it bodes well for the return on investment for security training. We will continue to explore this area further in future volumes to study this relationship further.

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 the application security fundamentals assessment should come as no surprise. Organizations are strongly encouraged to institute developer training and education programs to ensure a high competency level on application security. As Figure 36 shows there is an association between developers with better knowledge of application security fundamentals and improved security quality of applications. Hopefully readers will be able to utilize this data to both create a sense of urgency around developer training and justify its return.

7. Development agility and application security are not mutually exclusive

It is imprudent to have a conversation about software security devoid of any consideration of software development lifecycles. Contemporary development lifecycles such as agile or extreme advocate frequent "releases" in short development cycles (timeboxing) which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. The question we wanted to explore was, can software security be introduced into such modern development lifecycles without impact to the goal of speedy delivery of functionality. As Figure 11 shows the overwhelming majority (over 80%) of applications that failed to achieve acceptable security standards on initial submission and chose to remediate and resubmit, were able to pass within 1 week. This rapid turnaround time can easily be digested into any modern development lifecycle. This should serve to alleviate concerns around security slowing things down.

Recommendation: Conventional wisdom holds that integrating security into the software development lifecycle will result in more secure software. The data also suggests that this integration can be done with minimal impact on development lifecycles. Armed with this knowledge CIOs and CISOs can now more effectively counter concerns that might arise when initiating a formal application risk management program in their organization. We recommend that this formal approach be adopted for not just their internally developed applications but also those being procured from third-parties. Leveraging automation can be a cost-effective way to scale a formal program to all applications in an enterprise portfolio. Augmenting testing with appropriate training will also help tackle the issue more proactively by reducing the likelihood of introducing dangerous programming errors in the first place.

Software Supply Chain

If 2011 was the year of breaches it was also the year that left little doubt regarding the interdependence that an organization has on its software suppliers and service providers. A breach at email marketing service provider Epsilon[1] resulted in the compromise of millions of email addresses belonging to more than 40 organizations. Every organization's software infrastructure is comprised of applications they develop themselves and applications or components procured from third-party vendors. When examining the state of software security it is crucial to consider the influence of this extended software supply chain.

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. We continue to explore the 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. More than ever before we are seeing a demand for proof of independent verification being sought by CIOs and CISOs as part of their procurement or outsourcing processes. What's more is that this level of awareness is transcending beyond the security community with thought leaders and influencers in the sourcing and vendor management community calling for more third-party due diligence including code-level security analysis. (Refer to Forrester report, "Why Stronger Vendor Management is Essential to Managed Services Relationships" by analyst Jan Erik Aase.[2])

Awareness of risks inherited from the extended software supply chain goes beyond security community to also include call for action from thought leaders in sourcing and vendor management community.[3]

1 www.securityweek.com/massive-breach-epsilon-compromises-customer-lists-major-brands

2 www.forrester.com/rb/Research/why_strong_vendor_management_is_essential_to/q/id/59300/t/2

3 www.forrester.com/rb/Research/why_strong_vendor_management_is_essential_to/q/id/59300/t/2

Distribution of Application Development by Supplier Type

Figure 1 reveals similar statistics to Volumes 2 and 3 of the report. About 28% of the applications analyzed during the reporting period were identified as third-party (Commercial, Open Source and Outsourced vendors). As before, the large percentage of "internally developed" code likely reflects a data labeling issue, given the low percentage of outsourced applications (1%). Oftentimes the outsourced nature of certain applications labeled "internally developed" reveals itself upon remediation time when an outsourced party is assigned the task of fixing vulnerabilities. Hence we believe that the true percentage of "outsourced" code is higher than that represented in Figure 1.

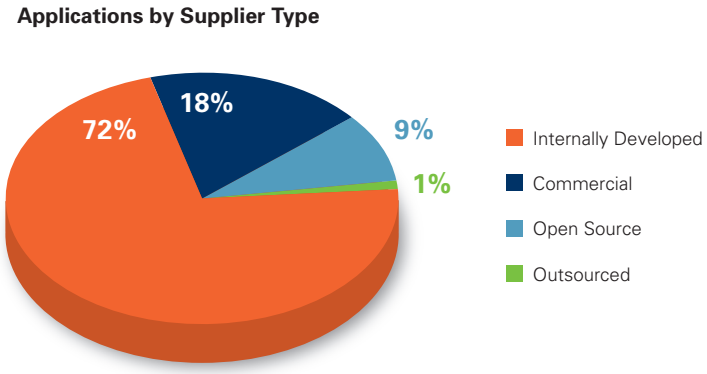

Figure 1: Applications by Supplier Type

The increase in the number of internally developed applications being assessed, from 70% to almost 72% may have been driven by an increase in the number of Veracode customers integrating automated scanning of their applications into their software development lifecycle and processes which is driving higher submission rates. In addition to that we are seeing an increased trend towards establishing a baseline of security quality across the entire application portfolio (or large swaths of it) by leveraging automated testing technologies and then determining where best to focus remediation efforts. This is also driving higher submission of new applications from within the same enterprise.

Distribution of internally developed vs. third-party applications within an enterprise portfolio remains similar to previous volumes—about one-third of applications are procured from third-parties with the remaining developed internally.

Distribution of Application Business Criticality by Supplier Type

As we have done since the inception of the State of Software Security report, we explored whether the business criticality level of an application had any bearing on the choice of software supplier. This is depicted in Figure 2. As before, a significant percentage of very high and high criticality applications are supplied by external suppliers (commercial, open source, or outsourced). The constant thread across all four reports has been that business criticality alone does not dictate the choice of an internal or external software supplier. 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.

Application business criticality not found to impact the choice of an internal or external software supplier.

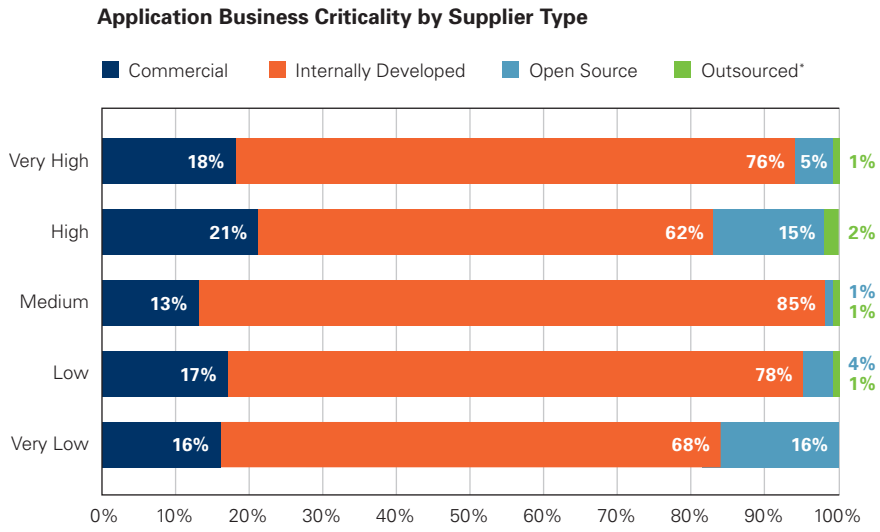

Figure 2: Application Business Criticality by Supplier Type (*Small sample size)

Distribution of Application Type and Programming Language by Supplier Type

Since Volume 3, Veracode has begun tracking some additional platforms. Mobile applications now account for a small but growing percentage of applications analyzed. At present, the sample size isn't sufficient to draw meaningful conclusions about the distribution of these platforms amongst supplier types so they have not been included in Table 1 (page 10). Many organizations are still determining how to best incorporate security testing into a markedly different development process.

Application Type and Programming Language by Supplier Type

	C/C++	ColdFusion	Java	.NET	PHP	Web	Non-Web
Internally Developed	4%	1%	60%	28%	6%	77%	23%
Commercial	15%	4%	48%	27%	7%	67%	33%
Open Source	21%	0%	55%	9%	13.3%	48%	53%
Outsourced*	2%	1%	60%	29%	7%	94%	6%

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

Most notable in Table 1 is the fact that C/C++ applications have decreased across all supplier types, reflecting that it is not a popular choice for new projects. However, since companies will never be able to deprecate their legacy C/C++ applications entirely, those applications must continue to be tested. The relative proportion of web applications to non-web applications has remained almost exactly the same, with only a few percentage points variation from Volume 3.

Application Security Performance Across Supplier Types

Figure 3 illustrates supplier performance on initial application submission as measured by the Veracode Levels standard. This is a significant change from prior versions of the report, which used the Veracode risk adjusted verification methodology to measure acceptability. This modification reflects a significant change that Veracode has made in its service offering (from which this data is gathered), which is to provide multiple yardsticks by which an application's security quality may be measured. The new standard is called Veracode Levels and is intended to provide a more stringent standard for measuring acceptability of application quality based on the changing threat landscape, and a capability for enterprises to define their own application risk tolerance. For example, the new policies disallow any occurrence of medium severity flaws (e.g. XSS) in Very High and High business criticality applications. More details about the Veracode Levels standard can be found in the Addendum.

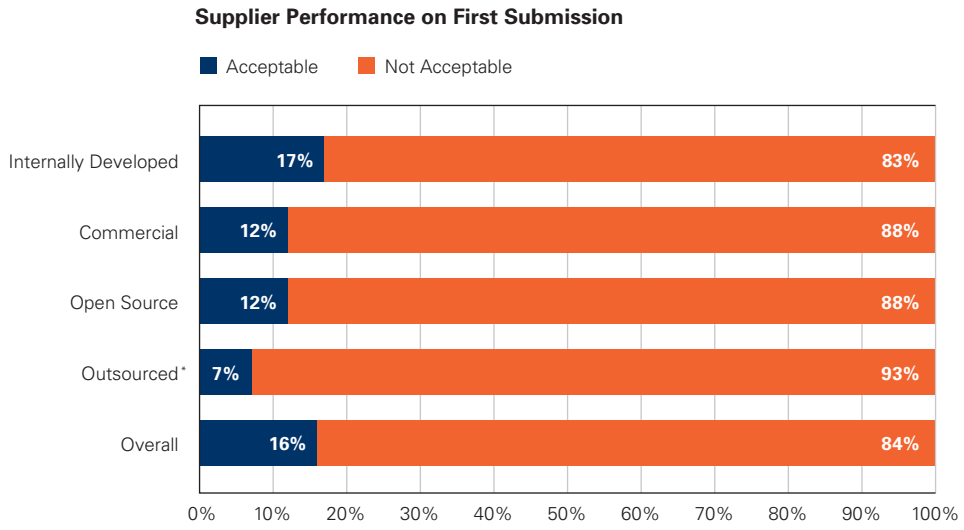

Figure 3: Supplier Performance on First Submission (*Small sample size)

When security standards are raised to a more stringent level more than 8 out of 10 all applications fail when first tested. Changes in the standard include zero-tolerance for any presence of SQL Injection and Cross-site Scripting for the highest criticality applications.

Figure 3 shows that a very small percentage of applications meet the more stringent test of acceptable security quality on first submission. Overall, only 16% of all applications were deemed acceptable. Across supplier types, the numbers varied but were all poor. Internally developed applications fared the best at 17% acceptable, followed by commercial and open source (tied at 12%). Outsourced applications had the lowest initial submission quality at 7%.

Distribution by Ability to Meet Security Compliance Policy by Supplier

Next we examine performance as measured against two industry standards—the OWASP Top 10 for web applications and the CWE/SANS Top 25 Most Dangerous Software Errors for non-web applications.

The OWASP Top 10 Compliance chart shows which non-web applications were deemed acceptable using the OWASP Top 10 list as a compliance policy. If even a single flaw for a category that is in the OWASP Top 10, such as Cross-site Scripting, is found, the application is deemed not acceptable. OWASP Top 10 compliance is one of the standards used by PCI DSS to determine if an application is secure enough to process credit card data.

Figure 4 shows OWASP performance for open source, commercial and internally developed applications. It appears that the percentage of applications that pass OWASP compliance for open source applications is less than half that of those passing for internally developed and commercial. One reason for lower compliance of open source is likely that most open source applications do not typically process credit cards or other sensitive data so there is less need to pass the OWASP policy.

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

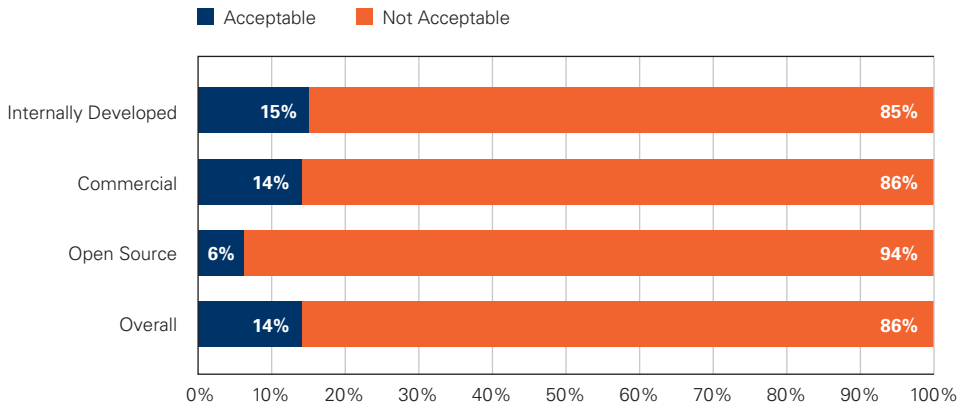

Figure 4: OWASP Top 10 Compliance by Supplier on First Submission (Web Applications)

The CWE/SANS Top 25 Compliance chart shows which non-web applications were deemed acceptable using the CWE/SANS Top 25 list as a compliance policy. If even a single flaw that is in a category contained in the CWE/SANS Top 25 list is detected in the application then it is deemed non-compliant.

In comparison to the OWASP policy compliance for web applications you can see right away that many more applications are meeting the CWE/SANS policy. Non-web applications meet this policy at a rate almost 3 times that of web applications meeting the OWASP policy. There are many more non-web applications that pass a strict policy than there are web applications even if the majority of non-web applications and web applications are non-compliant. The compliance by supplier for non-web applications follows a similar pattern as that for web applications. Open source fares worst, behind commercial, and internally developed.

Non-web applications perform significantly better against CWE/SANS Top 25 standard as compared to web applications against the OWASP Top 10 standard.

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

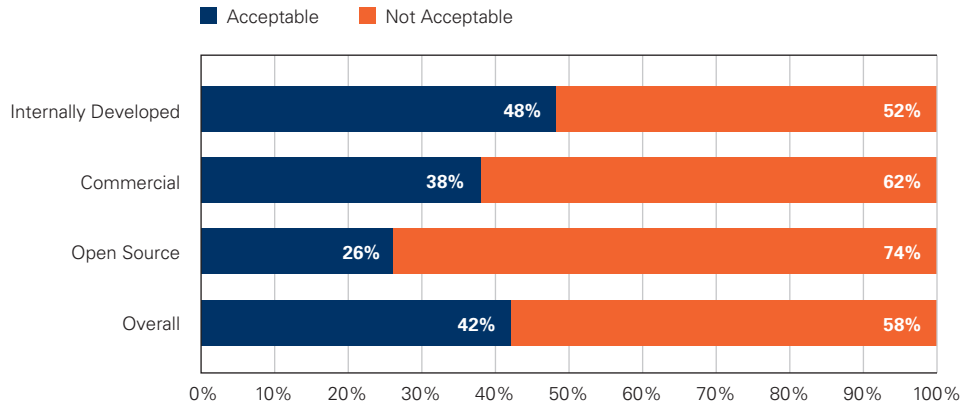

Figure 5: CWE/SANS Top 25 Compliance by Supplier on First Submission (Non-Web Applications)

Distribution of Most Common Security Vulnerabilities by Supplier

When looking at the vulnerability type distribution across different software suppliers some interesting patterns emerge. Some vulnerability types are prevalent across all suppliers at similar ranking. Some types are common but with differing ranking. Perhaps most interesting, some vulnerability types are unique to a supplier type.

The first commonality that stands out is Cross-site Scripting (XSS). This very prevalent vulnerability type is #1 for commercial, internally developed and open source. It is #2 for outsourced. All software supplier types are developing web applications and having difficulty preventing XSS flaws. This is a very easy to detect flaw using automated analysis techniques so we encourage readers to utilize static and dynamic analysis to find occurrences of this weakness in their software applications.

The second commonality is information leakage. This shows up as #2 for commercial and #3 for all the other supplier types. This vulnerability is often downplayed as low risk as by itself it rarely leads to a data breach. It is common however for several low risk vulnerabilities to be leveraged together in a single attack so it is prudent to take information leakage seriously.

Now let's focus on the issues that are unique to one software source. It is interesting that they all show up for commercial software. They are: buffer overflow, potential backdoor, and buffer management errors. None of these issues show up high in prevalence from other software sources. This is not to say other supplier types are completely free of these flaws, just that they are less prevalent. Buffer overflows and numeric errors are almost always found in non-typesafe languages such as C/C++ and ObjectiveC. These are the languages most heavily used by commercial software developers so the languages used are dictating the vulnerability mix. Since both of these vulnerability types have high impact when exploited and make it into the top 10 most prevalent issues for commercial software it behooves purchasers to test for these vulnerability types during the acquisition process.

There are a few reasons for why potential backdoors are solely in the top 10 list for commercial software. Potential backdoors are often remote support capability, phoning home for updates, or self modifying code. This by design but perhaps risky functionality is most often found in commercial software. Additionally since commercial software is most often compiled to binary instead of bytecode or interpreted it is much easier to hide malicious behavior.

Vulnerability Distribution by Supplier

Internally Developed		Commercial		Open Source		Outsourced*	
Cross-site Scripting (XSS)	58%	Cross-site Scripting (XSS)	44%	Cross-site Scripting (XSS)	41%	CRLF Injection	47%
CRLF Injection	12%	Information Leakage	11%	Directory Traversal	13%	Cross-site Scripting (XSS)	28%
Information Leakage	10%	CRLF Injection	8%	Information Leakage	13%	Information Leakage	6%
SQL Injection	4%	Directory Traversal	6%	CRLF Injection	11%	Encapsulation	6%
Cryptographic Issues	3%	Error Handling	5%	Cryptographic Issues	8%	Cryptographic Issues	5%
Encapsulation	3%	Cryptographic Issues	5%	SQL Injection	3%	Credentials Mgmt	3%
Directory Traversal	3%	Buffer Mgmt Errors	4%	Error Handling	2%	Directory Traversal	2%
Insufficient Input Validation	1%	Buffer Overflow	3%	Time and State	2%	API Abuse	1%
Time and State	1%	Potential Backdoor	3%	API Abuse	2%	Time and State	1%
Race Conditions	1%	SQL Injection	3%	Insufficient Input Validation	1%	Insufficient Input Validation	1%

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

Cross-site Scripting is the #1 vulnerability discovered across internally developed, commercial and open source applications. Buffer overflow, potential backdoors and buffer management errors far more prevalent in commercial software than other supplier types.

Remediation Analysis

One of the areas that sparks the most curiosity about our application security testing service is the post-analysis phase of the vulnerability management lifecycle. We get frequent questions from customers and analysts on what happens to the vulnerabilities once they are discovered, whether or not applications get remediated, and if so, how long does it take. In order to address these questions, we introduced the Remediation Analysis section in Volume 3 of the State of Software Security Report. We continue to report on the post-analysis remediation activities in this volume by exploring the behavior undertaken by different supplier types once vulnerabilities are discovered.

Figure 6 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%, 11-20%, … 91-100%) 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 91-100% of the commercial applications and slightly under 40% of companies in our dataset resubmitted 91-100% 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 91-100%, of their applications, it is actually mixed.

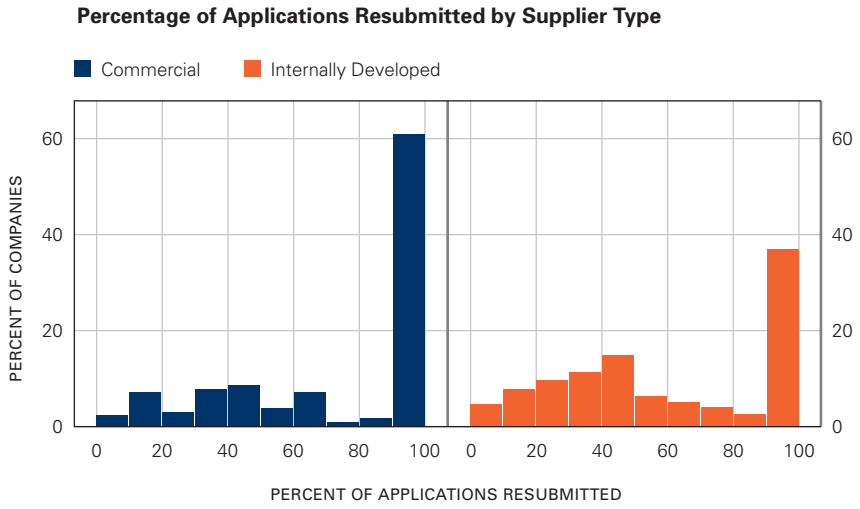

Figure 6: Percentage of Applications Resubmitted by Supplier Type

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 91-100% of the applications that need improvement.

Percentage of Applications Resubmitted by Business Criticality

As in Volume 3, we next performed the same analysis of resubmission activity broken down by application business criticality. The results are depicted in Figure 7. We continue to see a few similar patterns in the previous and current datasets.

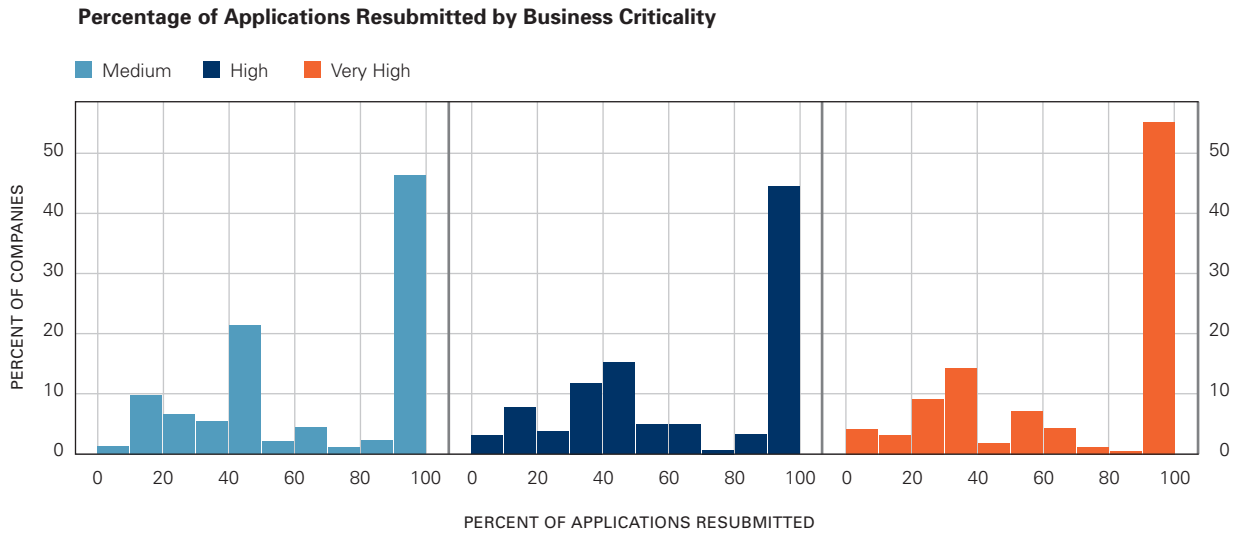

Figure 7: Percentage of Applications Resubmitted by Business Criticality

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 becomes more demanding as application criticality increases. This is particularly true for the acceptance criteria used in this report, as compared with previous reports. Third, the security quality is presumably more important as the business criticality of applications increases. What is surprising is that only slightly more than 50% of companies resubmit 91-100% 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 inconsistent with our expectation that remediation activity would mirror business criticality.

Percentage of Applications Resubmitted by Industry Type

Figure 8 shows an updated analysis of resubmission activity that is similar to that of Figure 7, but this time broken down by industry type for the Financial, Government, and Software verticals. The height of each bar reflects the percentage of companies that resubmit 0-10%, 11%-20%, …, 81-90%, and 91-100% of their applications for at least one additional review after the first.

This breakdown allows us to compare the remediation practices of the three industry sectors. The Government and Financial sectors share several important characteristics. They are both essential end users as opposed to developers of applications, they are both highly regulated, and they both are high value targets for exploits. In contrast, the Software sector is primarily a developer of applications and the products that it develops are often deployed in hundreds if not thousands of installations across many industry verticals.

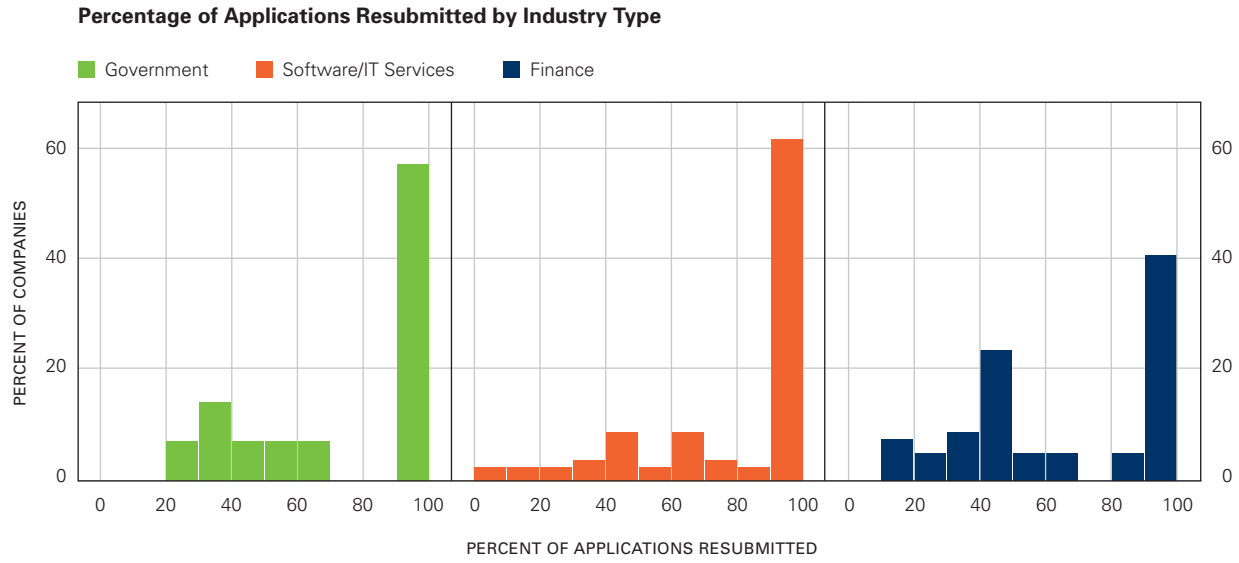

Figure 8: Percentage of Applications Resubmitted by Industry Type

On average, approximately 55% of submitters in the Government vertical resubmit 91-100% of their applications as compared to approximately 40% of those in the Financial industry vertical, and 60% of those in the Software vertical. The average provides a good estimate of the norm, but there is some variability amongst submitters. We will continue to explore this behavior by industry further.

Resubmission rates remain moderate across industry verticals, supplier type, and business criticality of applications. Highest resubmission rates occur for commercial applications within the Software/IT Services vertical market segment with very high business criticality.

Veracode Security Quality Score by Build

Much of this report details an application's security quality when first submitted for testing. This is the first snapshot we can take of an application and is the only time an application is not biased by our analysis findings and a remediation cycle. The Veracode Security Quality Score by Build shows the progress an application makes build over build as the developers respond to findings and attempt to remediate them.

In the last volume of this report we had the same whisker plot, but it only showed data for the first three builds. This was because we didn't feel we had enough data at that time to show more build scores. Previously, we saw a monotonic improvement in security quality score for each subsequent build. This shows that developers were fixing more security flaws than they were creating build over build.

In this volume, we have many applications for which we have analyzed 10 or more builds, so we have included data showing the progression of security quality score from build 1 to build 9. As you can see the score reaches its first peak at build 4, then the score drops for build 5. It increases for 2 more builds and then drops again for build 8. It increases again for build 9.

This oscillating behavior after achieving the first peak at build 4 suggests that building static analysis into an SDLC has a big effect on the first 3-4 builds and then enters into a maintenance mode where new security flaws are found as new code is written to introduce new functionality into the application. Only a few new issues are found and then are quickly remediated. So while the score oscillates it remains close to the score reached at build 3 or 4.

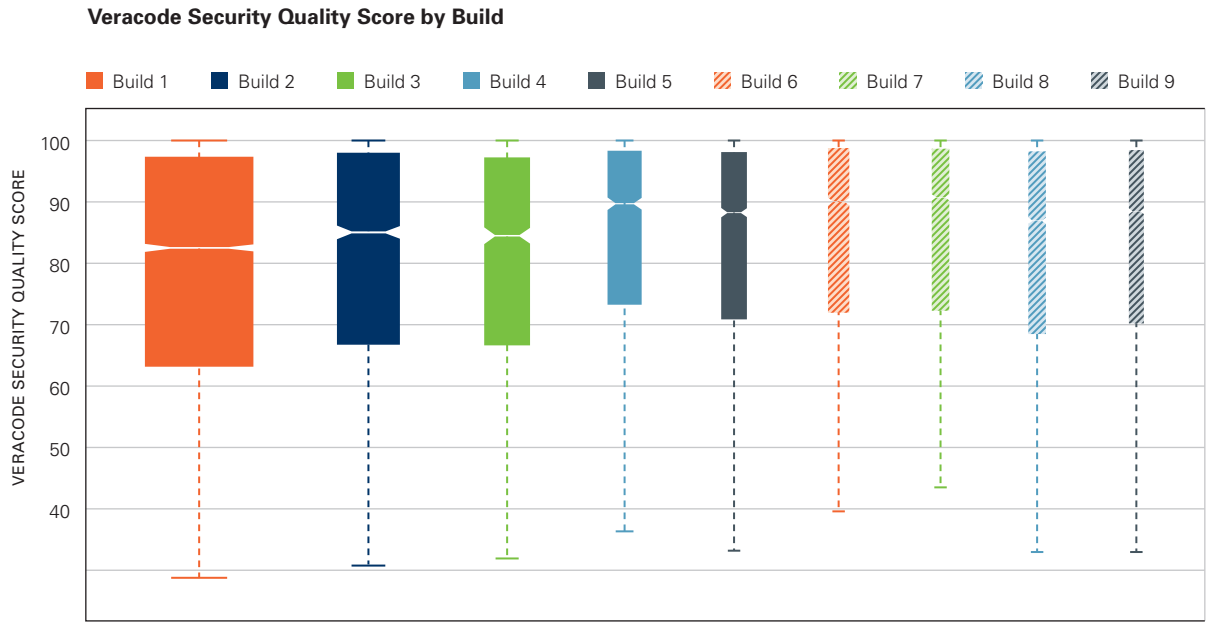

Figure 9: Veracode Security Quality Score by Build

Veracode Security Quality Score Trend by Quarter

Figure 10 provides trend data for mean Veracode Security Quality Score over the past eight quarters. The red line is the "best fit" via least squares regression to the set of plotted blue points, one blue point for each quarter. The slope of the red line is slightly smaller than that for the previous volume of this report and the p-value[4] has increased from .37 to .543. Both of these results are consistent as indicators that the trend is flat—no increase or decrease in application quality, as measured by the Veracode Security Quality Score, in our dataset since 4Q2009.

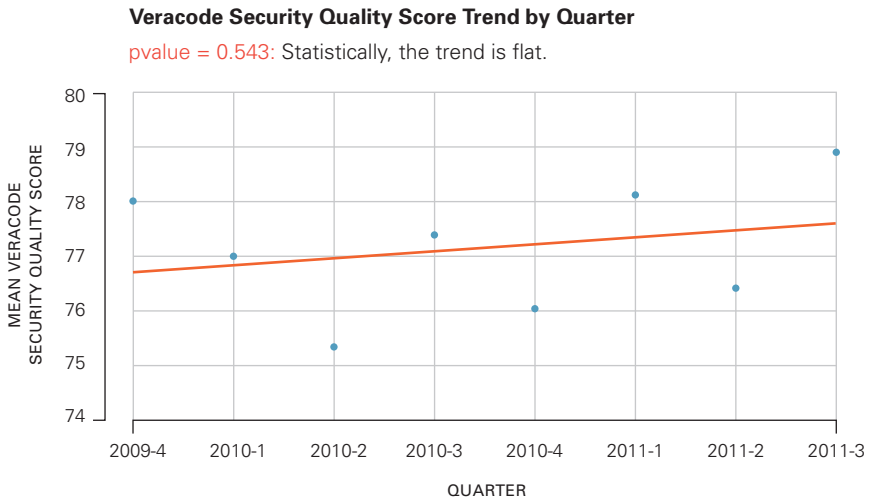

Figure 10: Veracode Security Quality Score Trend by Quarter

The workload mix for Veracode's service still contains a large proportion of new, previously untested applications which would tend to bias this result toward lower Software Quality Scores and an accordingly negative direction for the slope of the trend. However, applications with more mature SDLCs may be pulling the overall trend up, so that the resulting trend is flat. We will continue to monitor this trend as the workload mix represented by our dataset changes over time.

Security Quality Score trend remains flat over last quarters despite introduction of new previously untested applications with lower initial security quality. May be a sign that the part of the workload mix comprised of applications that are benefitting from security review embedded in their development process is pulling the overall trend up.

[4] For all of the trend graphs that are presented in this report, we also provide a p-value. In this context, the p-value can be viewed as a metric that is derived via a linear regression model. The p-value is the probability of observing a result at least as extreme as what we observed, given an assumption that the trend is flat. For example, in Figure 10, we observed what appears to be an increasing trend in Mean Veracode Software Quality Scores for eight quarters. A p-value of .543 indicates that the probability of seeing this in our data is over 54% when, in fact, the trend is flat--neither increasing or decreasing. This means that we can continue to be hopeful that the trend is up and we can continue to monitor progress, but, speaking statistically, the data does not support any conclusion other than that the trend is flat To look at a second example, in Figure 23, we see an apparent downward trend for the percent of affected web application builds. The p-value derived from this data is .048. This means that the likelihood that our observed trend is actually flat is only 4.8%—a very low probability. Thus, in this case we can state with 95% confidence that the trend is not flat and actually decreasing quarter over quarter, as it appears in the graph.

Time to Acceptable Security Quality

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 what is deemed acceptable. Figure 11 measures the time it takes get to acceptable security quality (as measured by Veracode Levels; see Addendum for description of this methodology). As readers may recall, in Volume 3 we had discovered that over 80% of applications that reach acceptable security quality across their lifespan do so within a period of 1 month. In this volume we decided to take a more microscopic view and analyze this data by weeks rather than months. As Figure 11 illustrates, the overwhelming majority (greater than 80%) of applications across supplier types achieve acceptable security quality within 1 week. This is very encouraging to see as it should serve to alleviate concerns around the effort level it may take to remediate critical vulnerabilities.

Note that in this dataset, 75% of all applications did not reach acceptable quality in the time period monitored in the report. Those applications are not represented in Figure 11. This large percentage may be explained by a few factors. We are witnessing a trend across organizations of using automated technologies to quickly baseline their entire application portfolio to gain an understanding of the weakest links in their software infrastructure. This results in a large number of applications being submitted and analyzed with remediation activities being focused on the most critical or the worst offenders first. Over time we expect to see other applications in the portfolio addressed and the percentage not achieving acceptability diminishing. We hope that the encouraging results on time to achieve acceptability security quality (over 80% overall within a week), drive a broader effort to remediate and improve security across the application portfolio.

For applications that achieve acceptable security quality do so within one week. This should serve to alleviate concerns around the timeframe it may take to remediate critical vulnerabilities post-analysis.

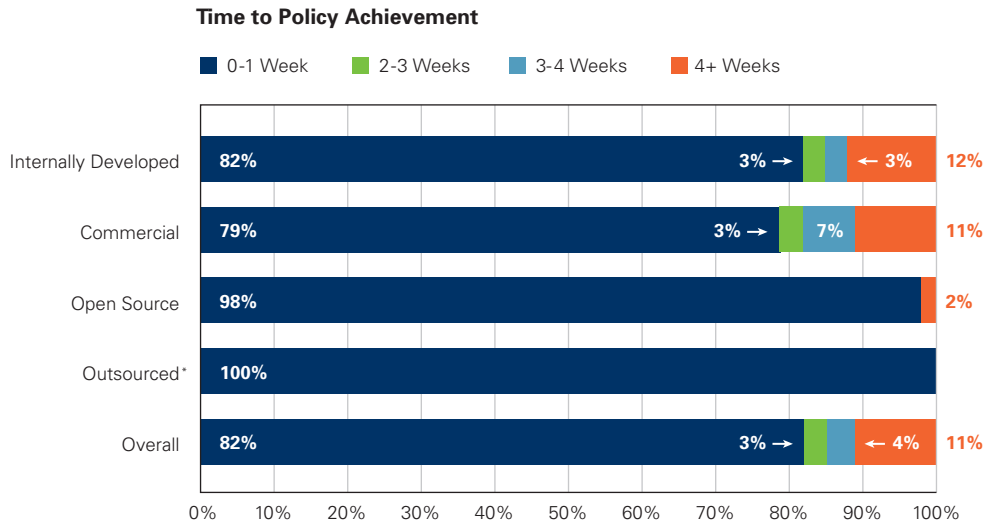

Figure 11: Time to Policy Achievement

Third-party Risk Assessments

Third-party risk assessments are defined as independent security assessments performed on third-party software using multiple testing techniques at the request of a buyer of that software. These buyers may be purchasing already developed applications for internal use, applications to be developed by someone else (e.g. outsourcers) and then used internally, or applications of either type to be re-distributed under an OEM, VAR, or other licensing arrangement. In some cases enterprises perform these assessments as part of their M&A due diligence process.

As we noted in earlier volumes, third-party assessments continue to be one of the fastest growing segments of our business and have contributed substantially to the growth of the dataset underlying this report. Recently this cause is being championed not just by security professionals but also by sourcing and vendor management professionals. (Refer to Forrester report, "Why Stronger Vendor Management is Essential to Managed Services Relationships" by analyst Jan Erik Aase.[5]) Most of the customers represented in this report have updated their procurement policies and contracts to include a requirement that third-party software undergo independent security verification prior to acquisition.

This section explores several aspects of this market including the types of enterprises that are engaging in this process, the types of applications that are most frequently being assessed, and the outcomes of the independent security verification process.

Requester Type by Industry

Figure 12 shows the types of organizations that are at the forefront of the third-party risk assessment market. These organizations typically integrate a formal policy into their procurement process requiring independent verification of the security quality of third-party software. As with Volume 3, 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, followed by Aerospace and Defense.

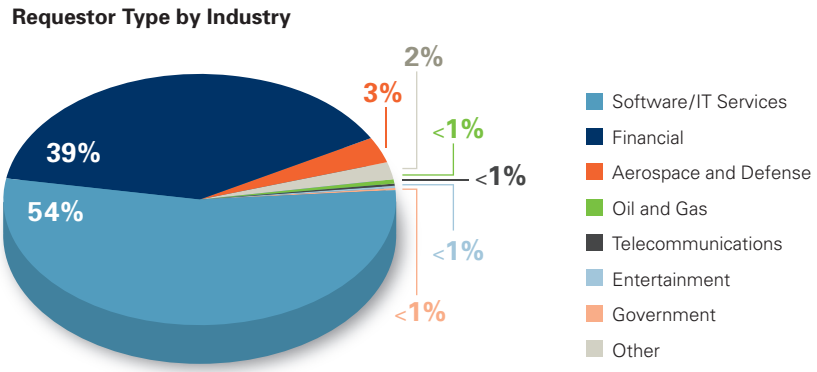

Figure 12: Requestor Type by Industry

[5] www.forrester.com/rb/Research/why_strong_vendor_management_is_essential_to/q/id/59300/t/2

The motivations for the two leading industries may be somewhat different. The Financial sector is heavily regulated with serious monetary and brand impacts if financial or customer data are compromised. Furthermore they recognize that the liability is still theirs, even if the source of the breach was a piece of third-party software (e.g. widely deployed commercial desktop applications). They need 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. They need to ensure that all participants in their platform ecosystem are providing secure code. Data also reveals a cross-industry trend for independent security verification of third-party software. There are a total of eight different industry segments represented by the enterprises engaging in this process. This is very encouraging to see.

Distribution of Third-party Assessments by Application Purpose

Next we examined the nature of the applications being analyzed by the above organizations. As with the last report, It appears that significant scrutiny is brought to bear by enterprises on both Operations and Financial applications. Typically these applications deal with critical information (such as credit card or other financial information) or support critical business operations. It stands to reason that an enterprise would want to gauge the security of these third-party applications before acquiring them. Similar factors may be at play when you consider the other categories of Learning and Growth, Customer Support, and Healthcare. All of these applications also handle sensitive data, making them reasonable selections for independent due diligence. Organizations that require these third-party assessments are clearly making application security a key element of their overall data protection strategy.

Choice of third-party application types subjected to independent verification signifies inclusion of application security testing as a key element of an enterprise's data protection strategy.

Third-party Assessments by Application Purpose

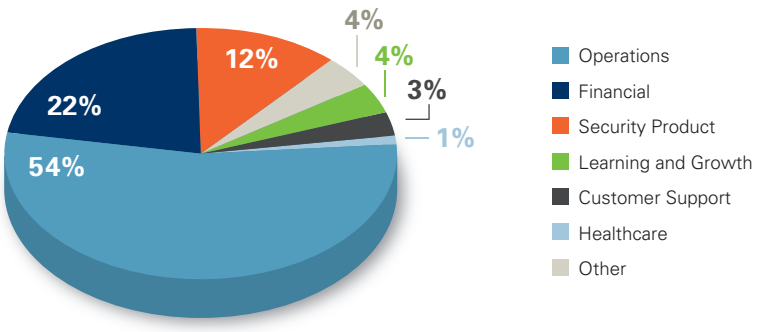

Figure 13: Third-party Assessments by Application Purpose

Performance Against Enterprise Policy Upon Initial Submission

In Volumes 2 and 3 we displayed performance of third-party applications against the Veracode risk adjusted verification methodology. As mentioned previously, we recently introduced a capability on our platform that allows enterprises to articulate their own application acceptance policies. As a result, we are now able to report on the performance of third-party applications against the enterprise determined policy. In other words, are third-party applications passing the threshold set forth by enterprises?

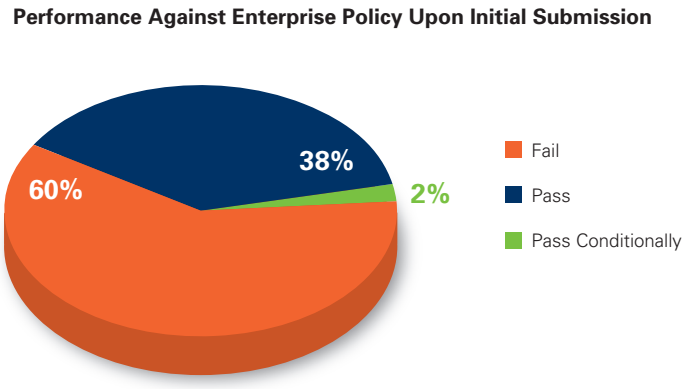

Figure 14: Performance Against Enterprise Policy Upon Initial Submission

As Figure 14 depicts, only 38% of third-party applications are compliant with enterprise policy when first tested, while 60% fail and 2% pass conditionally. To pass conditionally means that an application did not pass when first tested but still has a certain amount of time, called a grace period, to fix the offending flaws. The 60% of failing applications may eventually become compliant with the enterprise specified policy but did not do so within the time period included in this report. This high failure rate supports the concern that enterprises have regarding the software supply chain and explains their desire for independent security verification.

Nearly two-thirds of third-party applications fail to comply with enterprise dictated security policy when first subjected to independent security verification.

Performance Against Enterprise Policy by Application Purpose

We also explored how the performance against enterprise policy fared for different application types outlined in Figure 13. Figure 15 shows that Security Products, Operations and Healthcare had the worst compliance rates against enterprise policy when first tested. Enterprises may be subjecting these applications to greater scrutiny given their sensitivity.

Third-party Security Products, Operations and Healthcare applications suffer from worst compliance rates against enterprise specified security policies.

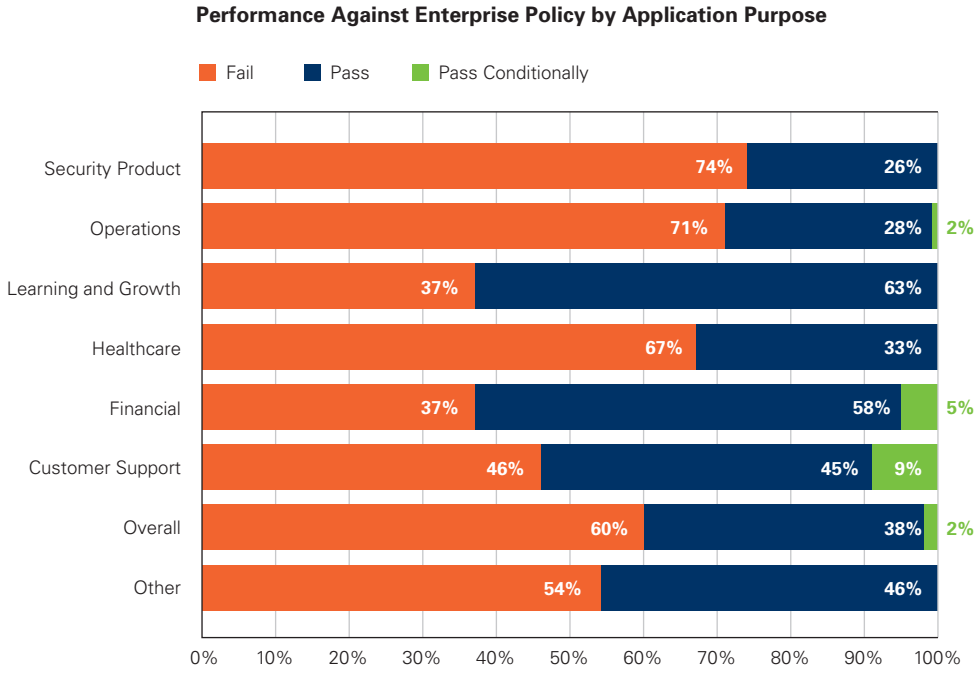

Figure 15: Performance Against Enterprise Policy by Application Purpose

For applications that were remediated, the time to become compliant with enterprise policy for more than 80% of the overall third-party dataset was still within 1 week. This should dispel fears that engaging third-parties for independent security verification and subsequent remediation activities is an overly arduous or time consuming initiative that will impact procurement timelines considerably.

More than 80% of applications that became compliant with enterprise policy did so within 1 week of initial test. This indicates that independent security verification of third-parties and subsequent remediation activities can be carried out fairly quickly.

Security of Applications

In the Security of Applications section we take a deep dive into the most prevalent vulnerability categories by volume and the vulnerability categories that affect the widest number of applications. We examine web and non-web applications separately because the prevalent vulnerability categories that affect these two groups are very different. This allows developers and security professionals to look at the data based on the type of applications they are concerned with. We also looked at trends over the past eight quarters to see if development teams are getting better or worse at preventing common 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. As was the case with Volume 3, three-quarters of the applications analyzed were web applications and a quarter were non-web applications.

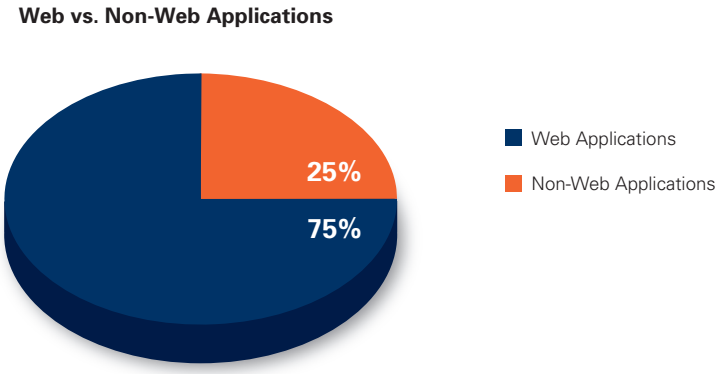

Figure 16: Web vs. Non-Web Applications

Distribution of Applications by Language

Figure 17 illustrates the language distribution of the applications in our dataset. As in our last report, there is a significant trend toward newer development platforms and away from C and C++; the latter comprises 9% of the sample in the current report compared to 12% of the prior report. Java and .NET have held roughly steady from the preceding report, while PHP has increased from 6% of the sample to almost 9%. The predominance of Java and .NET in the sample, together with the high proportion of web applications, suggests that enterprises continue to focus their security efforts largely on web applications. However, the growing prevalence of mobile platforms (Java ME, Android, and iOS together now comprise almost 2% of the total sample) suggests that enterprises are trying to stay abreast of new attack vectors as well.

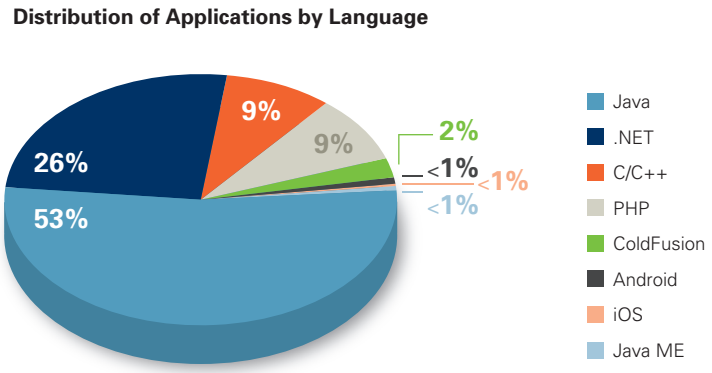

Figure 17: Distribution of Applications by Language

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 (2011) 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 overall frequency, accounting for 57% of all vulnerabilities in the dataset (Figure 18). This is an increase of 4% since our last report. The next three categories behind XSS—CRLF Injection, Information Leakage, and SQL Injection—differ slightly from our last three reports. CRLF Injection and Information Leakage have swapped places, and SQL Injection has surpassed Cryptographic Issues for the fourth spot. However, all of these movements were small and are probably insignificant.

In web applications, Cross-site Scripting (XSS) remains the most prevalent vulnerability category by overall occurrence frequency. In non-web applications Error Handling overtakes Buffer Overflow.

Top Vulnerability Categories
(Overall Prevalence for Web Applications)

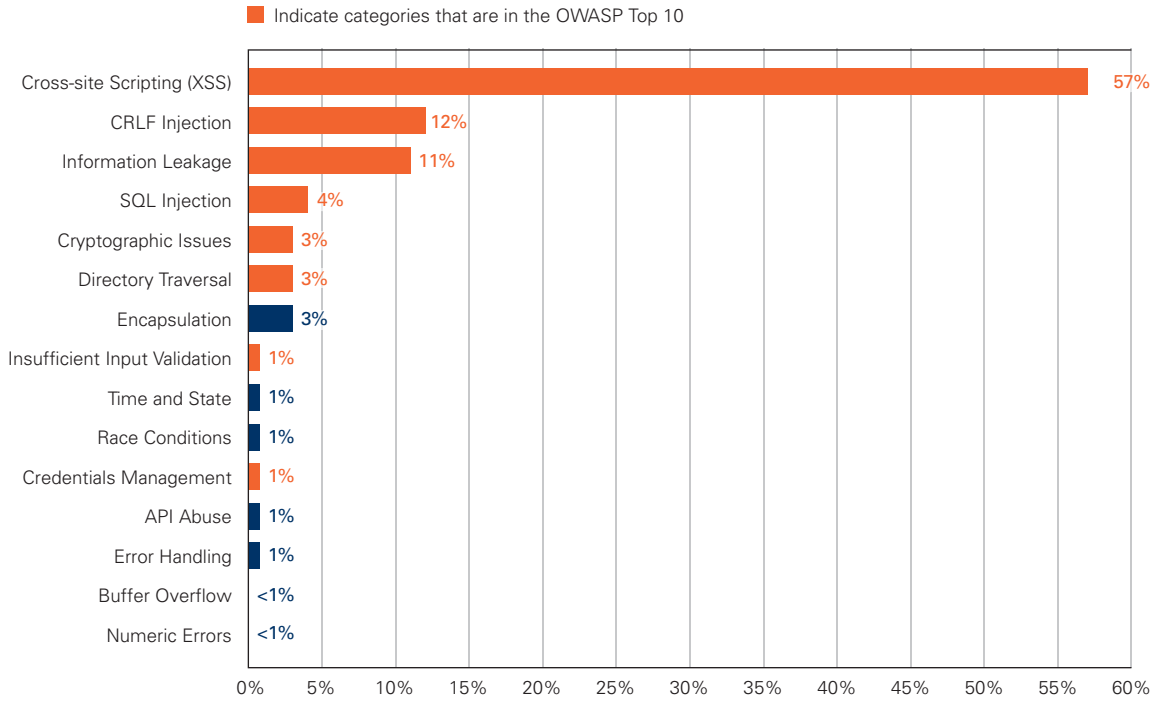

Figure 18: Top Vulnerability Categories (Overall Prevalence for Web Applications)

In non-web applications, Error Handling overtakes Buffer Overflow as the most prevalent category, accounting for 19% of all vulnerabilities detected. Buffer Management Errors experienced the largest shift, increasing to 15% from its previous share of 6%, edging out Buffer Overflows for second place. Still, Buffer Overflows clearly remain a significant issue. Numeric Errors also move into the top five, up from 6% to 9%.

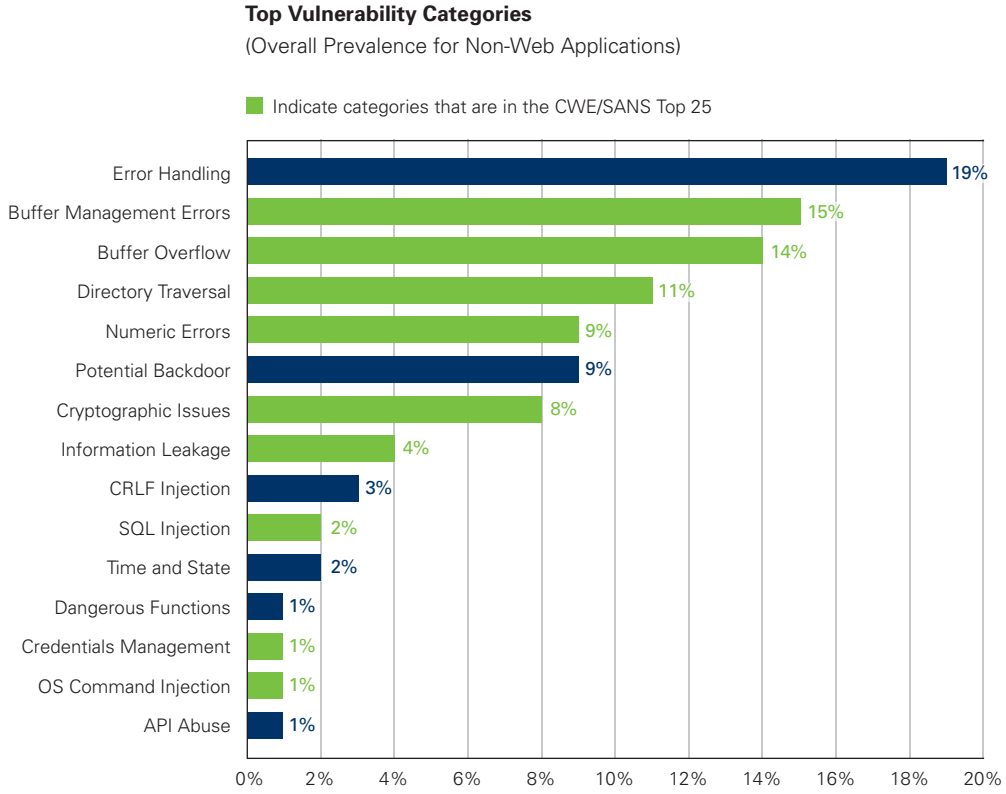

Figure 19: Top Vulnerability Categories (Overall Prevalence for Non-Web Applications)

The next set of charts show the percentage of applications affected by different vulnerability categories. One or more vulnerabilities in a category counts that application as having been affected by that vulnerability category. Since many applications have been analyzed across multiple builds, we are treating each application build as an individual unit to be counted. Therefore if an application has one SQL injection vulnerability in build 1 that build is counted towards the percentage of applications affected by SQL injection. If in build 2 that vulnerability is remediated and there are no other SQL injection vulnerabilities, then that build is counted towards applications not affected by SQL injection.

Figure 20 shows the top vulnerability categories for web applications only. The vulnerability categories that are listed in the OWASP Top 10 are highlighted in orange. The top six categories affecting web applications are also among the OWASP Top 10. The top category, Cross-site Scripting (XSS), affects 68% of all web application builds. We saw above that XSS was also the most prevalent by volume, comprising 57% of all vulnerabilities detected.

The perennial favorite of attackers, SQL Injection, affects about a third of all web applications analyzed. The most prevalent vulnerability by frequency, Cross-site Scripting impacts the largest number of web applications.

It is interesting to see that this flaw also affects the widest number of web applications. So it isn't just high counts of XSS in a minority of applications. This really is a pervasive vulnerability. SQL Injection is in sixth place affecting 32% of applications. This perennial favorite of attackers affects about a third of all web applications we analyze.

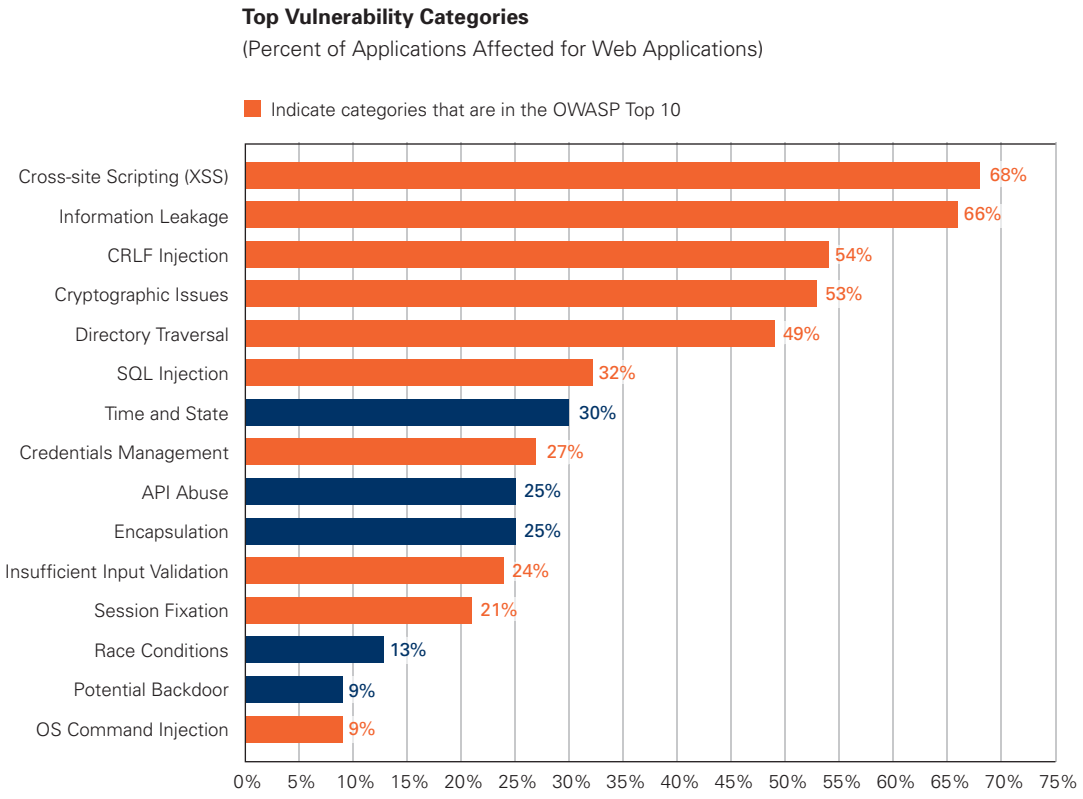

Figure 20: Top Vulnerability Categories (Percent of Applications Affected for Web Applications)

Figure 21 shows the top vulnerability categories for non-web applications only. In this chart the vulnerability categories that are listed in the CWE/SANS Top 25 are highlighted in green. Seven of the ten most prevalent categories overlap with the Top 25.

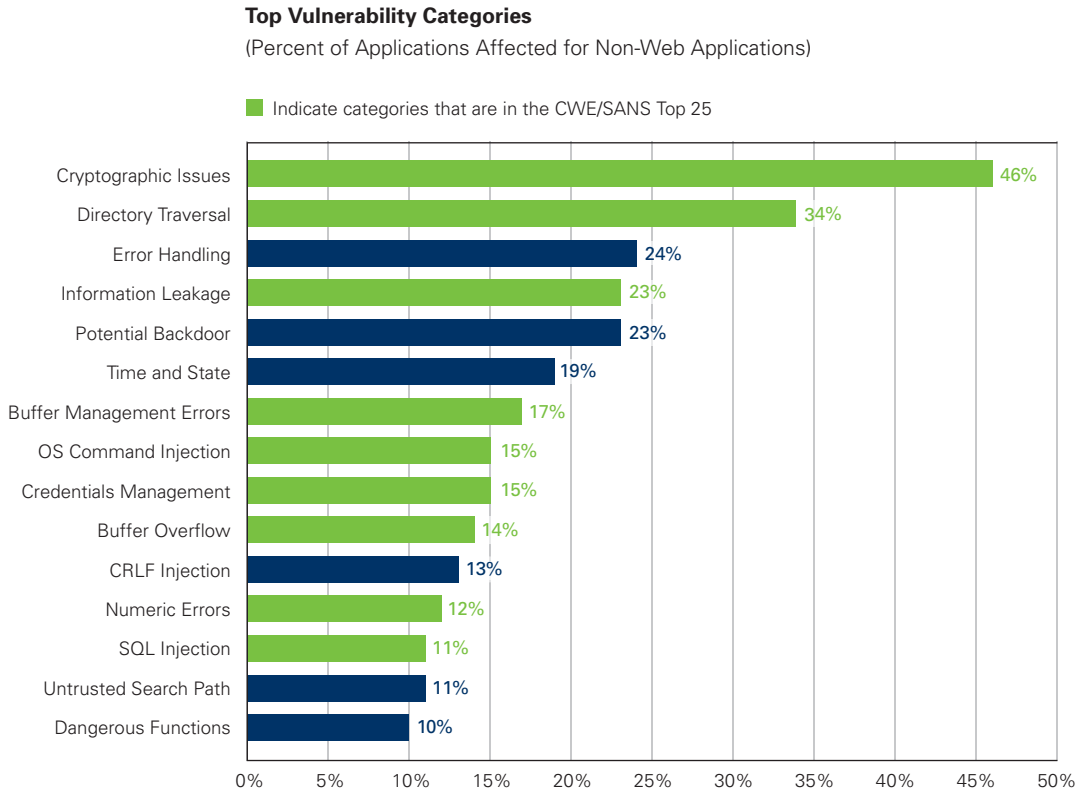

Figure 21: Top Vulnerability Categories (Percent of Applications Affected for Non-Web Applications)

Cryptographic Issues shows up on top, affecting 46% of applications. Above we saw Cryptographic Issues represented only 8% of all vulnerabilities we found. So while this vulnerability is not high in overall count it is still the most widespread across all non-web applications. Crypto functionality is common in non-web applications but is often not implemented securely.

Cryptographic Issues continue to impact nearly half of all non-web applications analyzed. Crypto functionality is common in non-web applications but is often not implemented securely.

Vulnerability Prevalence By Most Exploited Vulnerabilities

Some vulnerabilities matter more than others in the sense that they are more likely to be exploited. This may be due to the availability of automated attack tools or simply the fact that they are easy to detect. Looking at data published by the WASC Web Hacking Incidents Database (WHID), we extracted the 5 most prevalent application vulnerabilities that lead to web hacking incidents and mapped them to Veracode's prevalence data. The WHID list of Top Attack Methods[6] are as follows:

Vulnerability Prevalence (Percent of Web Applications Affected) by Most Exploited Vulnerabilities

Category	WHID	Mapping to Veracode Category	Web Applications Affected
Unknown	23%	Not enough information	
➤ **SQL Injection**	**20%**	**Maps to: SQL Injection**	**32%**
Denial of Service	12%	Not an application vulnerability	
➤ **Cross-site Scripting (XSS)**	**10%**	**Maps to: Cross-site Scripting**	**68%**
Brute Force	4%	Not an application vulnerability	
Predictable Resource Location	4%	Design flaw	
➤ **Unintentional Information Disclosure**	**3%**	**Maps to: Information Leakage**	**66%**
➤ **Credential/Session Prediction**	**2%**	**Maps loosely to: Cryptographic Issues**	**53%**
Stolen Credentials	2%	Not an application vulnerability	
CSRF	2%	Detectable only through manual testing	
Banking Trojan	2%	Not an application vulnerability	
Known Vulnerability	2%	Not enough information	
Misconfiguration	2%	Not an application vulnerability	
Process Automation	2%	Design flaw	
Abuse of Functionality	1%	Design flaw	
Content Spoofing	1%	Not an application vulnerability	
Administration Error	1%	Not an application vulnerability	
➤ **OS Commanding**	**1%**	**Maps to: OS Command Injection**	**9%**

Table 3: Vulnerability Prevalence (Percent of Web Applications Affected) by Most Exploited Vulnerabilities

[6] projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

Incidents in the WHID are caused by a variety of problems ranging from design flaws to configuration issues to application vulnerabilities. This section compares the issues caused by application vulnerabilities to the prevalence of those vulnerabilities (as measured by % of applications affected).

One might imagine that a vulnerability that affects the highest percentage of applications may also lead to the highest number of hacking incidents. However, the analysis above reveals no such direct correlation. SQL Injection, for example, affects about 32% of web applications and is also one of the most common vulnerabilities exploited in web hacking incidents (20%). However, OS Command Injection (WASC calls it OS Commanding), on the other hand, does not exhibit the same pattern. Even though this category is particularly severe, affecting 9% of web applications and allowing an attacker to execute arbitrary commands on the server, it only accounts for 1% of the web hacking incidents. At first glance, this data might suggest that it's acceptable to postpone remediation of OS Command Injection vulnerabilities, since it only accounted for a small percentage of reported incidents. But considering that it has such serious security ramifications, it should continue to be prioritized highly for remediation. Indeed, it is wise to consider the selection biases in both sets of data before making any specific remediation prioritization decisions.

Similar to CVE, the WHID only represents hacking incidents that were publicly disclosed. It's a fair assumption that companies aren't reporting these incidents unless they are required to do so by law or to as part of a compliance framework. So for example, there very well could be a rash of OS Command Injection hacks that were quietly swept under the rug and never counted in the WHID. Similarly, the WHID data can't possibly include attacks that were never detected in the first place.

The selection bias in the Veracode data, as we've pointed out before, is that these are the applications that are deemed important enough by their respective organizations to warrant scanning in the first place. In other words, on a sliding scale of business criticality, most of these are in the upper half. Attackers don't necessarily need to target the critical applications directly; they can start with the old crusty applications that the organization has forgotten about, and work their way deeper into the infrastructure from that point. In other words, while there may not be direct correlation between the WHID and Veracode categories, it is instructive to examine the number of applications afflicted by vulnerabilities that have been known to cause real incidents.

Based on the fact that there are so many unknowns about how these datasets relate to one another, we believe it is most prudent to not just to focus on vulnerabilities that cause incidents but rather a combination of that data along with vulnerability prevalence to prioritize remediation activities.

Quarterly Trending by Vulnerability Type

The quarterly trend for number of applications affected by Cross-site Scripting (XSS) was flat at 60% over a two year period in Volume 3. In this volume the trend continues to remain statistically flat, based on a p-value of 0.124. The third quarter of 2011 shows 70% of applications affected. Eradicating XSS has turned out to be a great challenge even though this vulnerability category is easy to find and remediate. Many development teams downplay the risk associated with XSS so often these vulnerabilities don't get fixed. It may behoove them to remember that 1 in 10 publicly reported web hacking incidents are caused by this vulnerability type (as indicated by the WASC Web Hacking Incident Database discussed earlier).

Good news on SQL Injection—Downward trend continues with 32% of web applications containing one or more SQL Injection issues this quarter as compared to 38% this quarter 2 years ago.

However, there is good news for another prevalent and exploited vulnerability: SQL Injection is trending downward. The downward trend we saw in our last report has continued through 2011. In the fourth quarter of 2009 we saw 38% of applications affected and now we see in the third quarter of 2011 only 32% of applications effected. The highly publicized data breaches caused by SQL injection and the resulting education around the issue has undoubtedly put pressure on development teams to work harder at eradicating this vulnerability class. This is a very heartening trend. Perhaps in 10 years SQL Injection will be extinct, 23 years after being discovered.

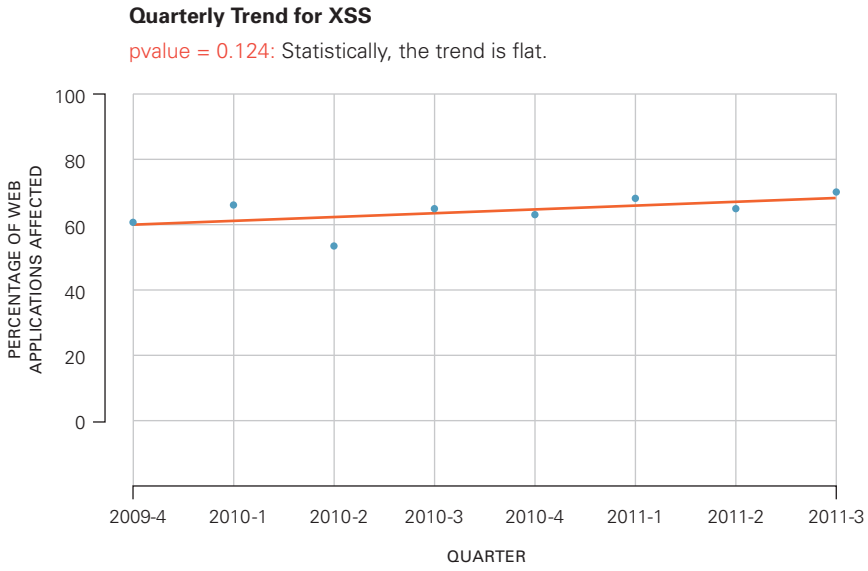

Figure 22: Quarterly Trend for XSS

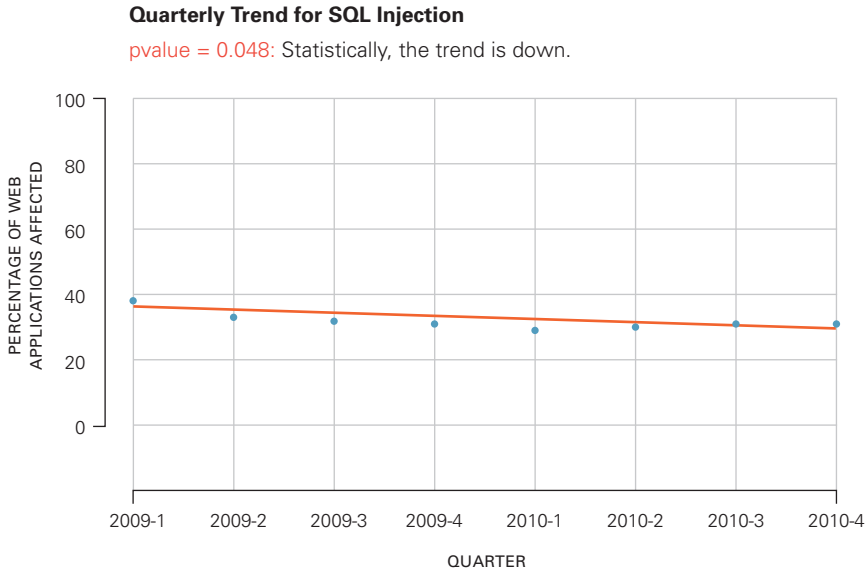

Figure 23: Quarterly Trend for SQL Injection

Vulnerabilities by Language Distribution

Table 4 (page 35) presents the most prevalent categories (by share of total vulnerabilities discovered) by language family. New to this volume of the State of Software Security report is the addition of data from the analysis of applications written for the Android mobile platforms. These applications were coded in Java, so it is unsurprising to see common Java flaws like cryptographic issues, CRLF injection, directory traversal, and time and state management on the top flaw list. More troubling was the high prevalence of information leakage flaws, referring to specific cases where the application was transmitting potentially sensitive data using one of the many communication mechanisms available on the phone. These flaws accounted for 10% of the flaw volume in Android phones and a full 38% of the flaws found on BlackBerry phones. This is a concern because of the sensitivity of the data typically carried on smartphone platforms (location data, personal information like contacts, email, or voicemail, camera and voice data) and because of the multitude of communication options available. The prevalence of information leakage on the Android platform is particularly interesting given the prevalence of third party advertising libraries in these applications (see Veracode's blog posts on information leakage in the Pandora Android application for more context[7]).

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 greater awareness of the need to prevent XSS.

[7] www.veracode.com/blog/2011/04/mobile-apps-invading-your-privacy/

Vulnerability Distribution by Language

Java	ColdFusion	C/C++	.NET	PHP	Android	Java ME
Cross-site Scripting (XSS) **56%**	Cross-site Scripting (XSS) **87%**	Error Handling **26%**	Cross-site Scripting (XSS) **47%**	Cross-site Scripting (XSS) **75%**	Cryptographic Issues **44%**	Cryptographic Issues **58%**
CRLF Injection **16%**	SQL Injection **8%**	Buffer Overflow **20%**	Information Leakage **18%**	Directory Traversal **10%**	CRLF Injection **28%**	Information Leakage **38%**
Information Leakage **10%**	Directory Traversal **1%**	Buffer Mgmt Errors **18%**	Cryptographic Issues **10%**	SQL Injection **7%**	Information Leakage **10%**	Directory Traversal **3%**
Encapsulation **4%**	Information Leakage **1%**	Numeric Errors **14%**	Directory Traversal **9%**	Information Leakage **4%**	Time and State **8%**	Time and State **1%**
Cryptographic Issues **3%**	CRLF Injection **1%**	Potential Backdoor **14%**	SQL Injection **6%**	Code Injection **2%**	Cross-site Scripting (XSS) **4%**	

Table 4: Vulnerability Distribution by Language

The other major web vulnerability of concern is SQL injection. This vulnerability is of particular concern given the rash of data loss incidents and of public data theft by "hacktivist" groups such as Anonymous and Lulzsec, many of which exploited SQL injection flaws as a means to access sensitive data and to embarrass their intended targets. As in Volume 3, SQL injection was prevalent in PHP (rising from 6% to 7%) and ColdFusion (falling slightly from 9% to 8%), but a significant change was the inclusion of SQL injection in the top five .NET vulnerability types, accounting for 6% of all discovered issues. Given the announcements in October 2011 of mass exploitation of SQL injection vulnerabilities in ASP .NET applications, this increase is of particular concern.[8]

SQL Injection rises to be in the top 5 .NET vulnerabilities. Potential backdoors are uniquely popular to C/C++.

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.

[8] Chickowski, Erica. "Mass SQL Injection Attack Hits 1 Million Sites." Dark Reading. www.darkreading.com/database-security/167901020/security/news/231901236/mass-sql-injection-attack-hits-1-million-sites.html (2011-10-19)

Distribution by Ability to Meet Security Compliance Policy by Industry

The next two figures provide insight regarding the security of applications by industry segment on initial review as measured against the OWASP policy for web applications and CWE/SANS policy list for non-web applications. The OWASP Policy assigns an acceptable rating to a web application if it contains no flaws in the 2010 OWASP Top 10 list of web application vulnerabilities. The SANS Top 25 Policy assigns an Acceptable rating on a non-web application if it contains no flaws in the 2011 SANS Top 25 list of application vulnerabilities.

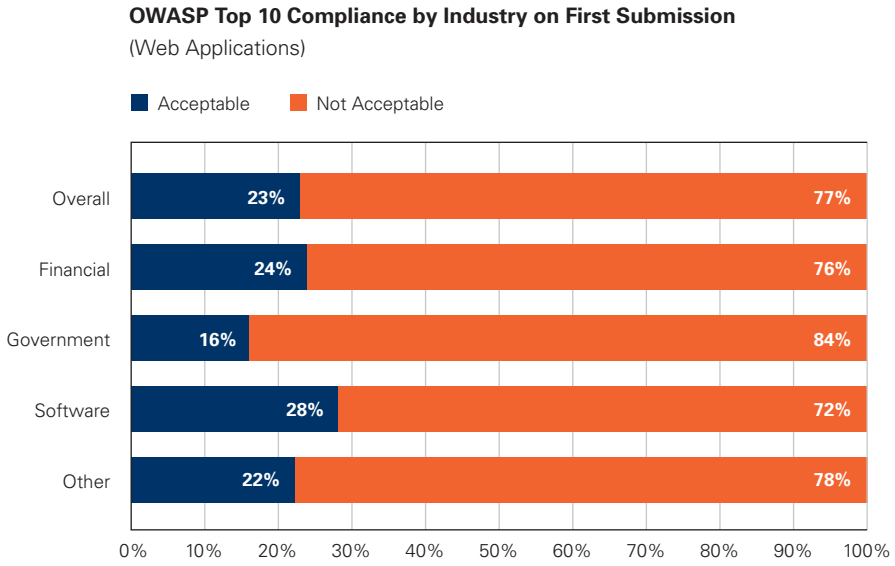

Figure 24: OWASP Top 10 Compliance by Industry on First Submission (Web Applications)

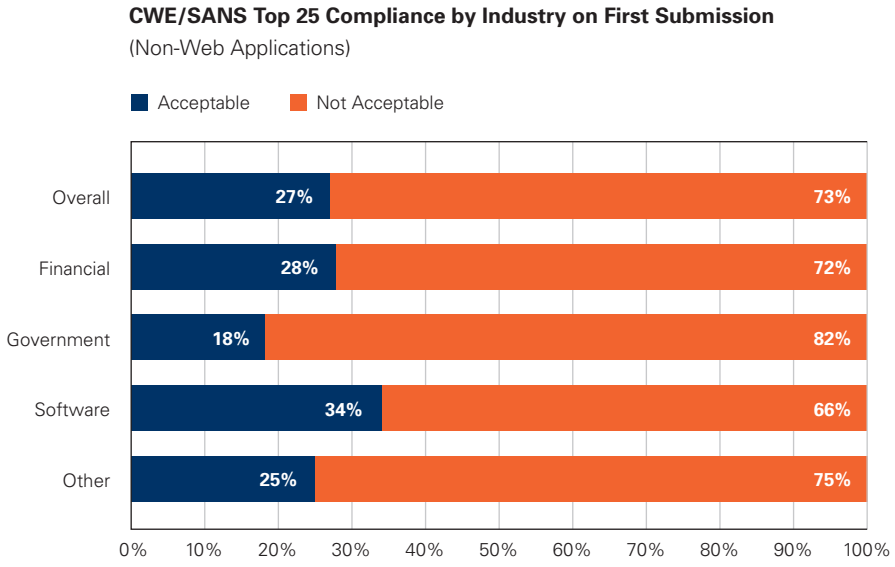

Figure 25: CWE/SANS Top 25 Compliance by Industry on First Submission (Non-Web Applications)

Examining Figures 24 and 25 (page 36) it appears that the Software Industry has the best performance against both the OWASP policy (for their web applications) and the CWE/SANS policy (for their non-web applications). Financial Industry follows suit with the Government sector trailing the pack. We have undertaken a deeper analysis of the Government sector relative to other industry segments later in this report that shed some light on their poor performance. Readers should note that despite the rankings depicted the difference in performance against the industry sectors is not significantly wide and there is much room for improvement against these standards across the board.

Android Application Analysis

Mobile malware has been on the rise in 2011, with the Android Market having been hit disproportionately more than Apple's App Store or BlackBerry App World. In this section we present a sneak peek at some data we've collected from a small set of Android applications that we've scanned (under 1% of our dataset). Due to the limited amount of data, we expect the statistics illustrated here to experience some variability over the next few reporting periods till the dataset grows to a larger number.

Figure 26 shows the distribution of Android applications submitted by different industry verticals. Retail and Technology combined to account for over half of Android applications, with a significant chunk coming from Financial Services, Banking, and Media. It is not surprising to see these verticals at the top of the distribution due to the sensitivity of the data and transactions their mobile applications are processing.

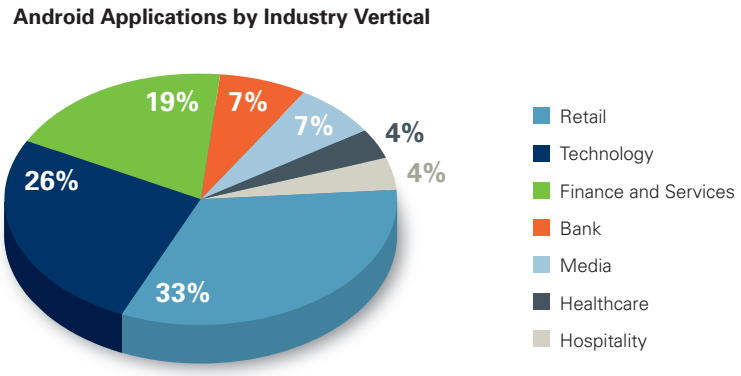

Figure 26: Android Applications by Industry Vertical

Our initial areas of focus for mobile applications have been Information Leakage (i.e. data exfiltration: transmitting potentially sensitive information off the device) and Cryptographic Issues. Table 5 shows the prevalence of these flaw categories across the Android applications analyzed. Each of these categories has a couple of different CWEs represented within it.

Android Percentage of Applications Affected (For Top CWE Categories)

Flaw Category	CWE Category	CWE	Percent Applications Affected
Cryptographic Issues	Insufficient Entropy	331	61%
Cryptographic Issues	Use of Hard-coded Cryptographic Key	321	42%
Information Leakage	Information Exposure Through Sent Data	201	39%
Information Leakage	Information Exposure Through Error Message	209	6%

Table 5: Android Percentage of Applications Affected (For Top CWE Categories)

With regard to Cryptographic Issues, 61% of Android applications exhibited at least one instance of Insufficient Entropy. In Java applications this usually takes the form of using the statistical random number generator (RNG) rather than the cryptographic RNG. It's a common mistake seen frequently in Java web applications and can be fixed with a single line of code. Even more interestingly, 42% of Android applications contained a hard-coded Cryptographic Key. Compared to the full dataset of non-Android Java applications, only 17% of them contained a hard-coded Cryptographic Key. Ironically, a hard-coded key is much simpler to extract from a mobile application than from a J2EE web application since the application can simply be copied off the mobile device! This category will be worth watching to see if the prevalence gap persists as the Android population grows.

Under Information Leakage, nearly a third of Android applications were transmitting at least one piece of information marked as potentially sensitive. However, measuring the security of mobile applications is dependent on understanding the gap between design and implementation. It's not possible to determine whether or not these info leaks were intentional, or if the observed behaviors would have been obvious to the user based on the nature of the application. For example, is it a privacy leak if an application is transmitting your GPS location? If the application is FourSquare, probably not; sharing location information is core to its functionality. On the other hand, if the application is a solitaire game, then the use of GPS data may signify malicious behavior.

In the future, we hope to incorporate metadata about the application in combination with various heuristics in order to gain additional insight into questionable application behaviors.

Government Sector Analysis

Readers may recall that in Volume 2 we explored the financial industry in more depth and in Volume 3 we delved into the software industry segment. In this report we conduct a deeper dive into the Government sector and compare various aspects of Government applications to those in the Finance and Software sectors. With half a dozen or more cybersecurity bills in various stages of the legislative process it is clear that lawmakers are finally turning their attention to the protection of the nation's information infrastructure. These proposed legislations are diverse in the areas they cover. Some serve to promote public awareness of cyber security (e.g. S.813 proposed by Senator Sheldon White-house) while others look to enhance the security and resiliency of the cyber and communications infrastructure of the United States (e.g. S.413 proposed by Senator Joseph Lieberman, which includes specific language for assessing risks from the software supply chain). It is instructive to take a closer look at Government applications as well as carry out a comparative analysis to better inform these timely debates on regulations that serve to increase information security across the board.

Percentage of Applications by Language Family (Government)

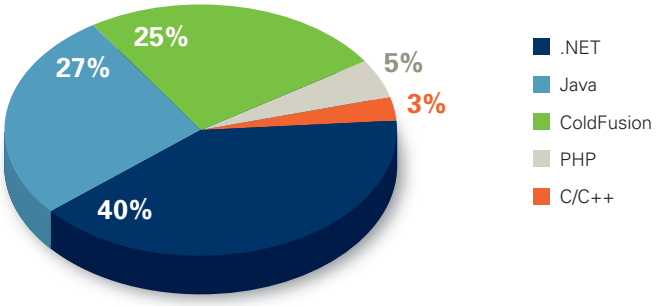

Figure 27: Percentage of Applications by Language Family (Government)

Figure 27 depicts the relative proportion of Government Applications by language family. The following table provides a head to head comparison of the distribution of Government vs. the entire population of applications in this dataset.

Distribution of Government Applications by Language Family vs. Overall Application Distribution by Language Family

	Government	Overall
.NET	40%	26%
C/C++	2%	9%
ColdFusion	25%	2%
Java	27%	53%
PHP	5%	9%

Table 6: Distribution of Government Applications by Language Family vs. Overall Application Distribution by Language Family

Government applications make significantly higher use of ColdFusion than the overall dataset. .NET comprises nearly half of all Government applications.

There are significant differences between the distribution of Government applications and that of the entire population of applications in our dataset. Government applications' use of ColdFusion outstrips the general population by a factor of 10. Government applications leverage ColdFusion almost as much as Java. This is partially explained by the fact that the vast majority of our Government applications dataset is comprised of web applications where ColdFusion is a popular programming language.

Percentage of Government Applications by Business Criticality

Figure 28 shows the distribution of government applications by supplier type across applications of different criticality. As with the overall dataset there is reliance on third-party suppliers for Very High, High and Medium business criticality applications. In fact, a greater percentage of applications at these criticality levels are procured from commercial vendors for Government than the overall dataset (e.g. 24% of Very High criticality government applications are commercial vs. 18% for the overall dataset). This further increases the importance of independent verification of software applications procured from outside vendors for Government. In recent years OMB has directed agencies toward (Commercial-Off-the-Shelf) COTS software vs. costly internally developed applications wherever possible to drive standardization, lower cost and timely implementation.[9] This trend is reflected in the mix of government applications submitted for security reviews and the growth in requests from the Government sector for third-parties to submit COTS software for analysis.

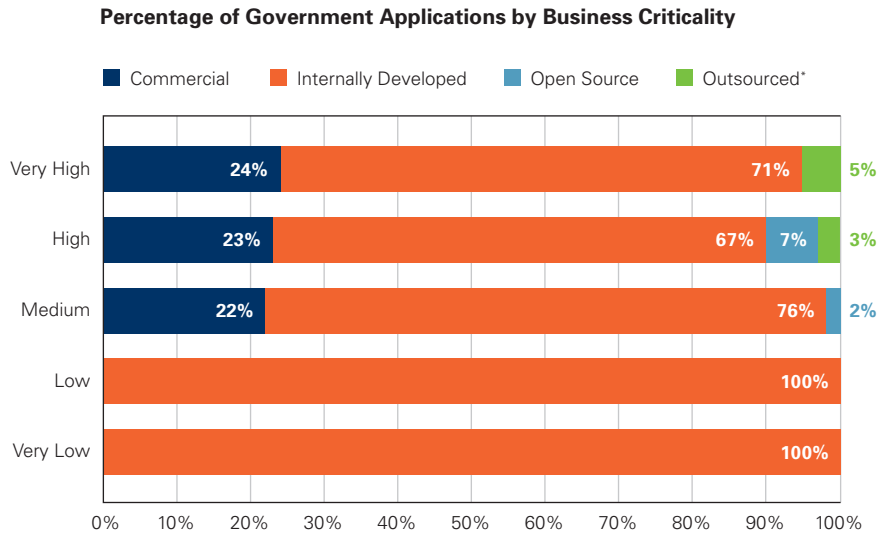

Figure 28: Percentage of Government Applications by Business Criticality

[9] www.whitehouse.gov/omb/circulars_a130_a130trans4

Comparison of Government Web Application Vulnerability Prevalence vs. Other Industry Sectors

Table 7 provides a tabulation of the percent of affected web applications by vulnerability category and industry type. The percent of affected web applications is computed for each vulnerability category as the count of applications in which at least one instance of that vulnerability category was discovered over the total number of applications associated with the industry type of the application. Information Leakage and Cross-site Scripting (XSS) flaws rank among the top three categories that affect web applications across all industry types and affect over half of the web applications in our dataset.

Three-quarters of all Government web applications contain Cross-site Scripting issues and 4 in 10 contain SQL Injection issues. This is a higher incidence rate than in Finance and Software.

Cross-site Scripting is interesting in that it affects a higher percentage of Government applications (75%) than applications from the other industry sectors (67% for Finance and 55% for Software). A partial explanation of this maybe offered by the fact that Government applications utilized a higher percentage of ColdFusion than other industry segments. As Table 4 (page 35) demonstrates ColdFusion has a higher incidence of XSS issues as compared to other platforms. ColdFusion also tends to be used by less experienced developers for creating web applications with greater ease. These developers are also less likely to be experienced in secure coding practices. Similarly, we found that 40% of government web applications had SQL Injection issues as compared to 29% for Finance and 30% for Software.

Comparison of Government Web Application Vulnerability Prevalence vs. Other Industry Sectors

Government ▼		Finance		Software	
Cross-site Scripting (XSS)	75%	Information Leakage	68%	Cryptographic Issues	59%
Information Leakage	66%	Cross-site Scripting (XSS)	67%	Information Leakage	59%
SQL Injection	40%	Cryptographic Issues	53%	Cross-site Scripting (XSS)	55%
Cryptographic Issues	35%	CRLF Injection	51%	CRLF Injection	54%
Directory Traversal	31%	Directory Traversal	47%	Directory Traversal	54%
Insufficient Input Validation	27%	Insufficient Input Validation	30%	Time and State	39%
CRLF Injection	27%	SQL Injection	29%	Credentials Mgmt	31%
OS Command Injection	19%	Time and State	28%	SQL Injection	30%
Time and State	18%	API Abuse	26%	API Abuse	25%
Credentials Mgmt	16%	Encapsulation	25%	Encapsulation	23%
API Abuse	14%	Credentials Mgmt	24%	Session Fixation	18%
Potential Backdoor	12%	Session Fixation	19%	OS Command Injection	14%
Session Fixation	11%	Race Conditions	13%	Potential Backdoor	14%
Encapsulation	11%	Potential Backdoor	10%	Race Conditions	13%
Untrusted Search Path	3%	OS Command Injection	6%	Insufficient Input Validation	13%

Table 7: Comparison of Government Web Application Vulnerability Prevalence vs. Other Industry Sectors

Given the prevalence of Information Leakage, Cross-Site Scripting, and SQL Injection vulnerabilities in Government web applications, it is interesting to see if the trend in percent of affected web applications is improving. The next three figures, Figures 29, 30, and 31 depict the trend in percentage of affected web applications for each of these vulnerability categories over the past two years (8 quarters).

In each of Figures 29, 30 and 31, we plot the percentage of affected Government web applications, by quarter, and then display a best-fit red line determined via a regression least squares fit model. A line with negative slope would indicate possible improvement. Parameters from the regression model indicate the confidence level with which we can state that the indicated trend is indeed up or down. In all three cases, we cannot conclude with sufficient confidence (95% or higher) that the trend is either up or down. Statistically, each of the trends is flat, that essentially the percentage of affected web Government applications has not changed over the past two years for Cross-Site Scripting, SQL Injection, and Information Leakage vulnerabilities. This is discouraging because of all the attention that has been devoted to these three high visibility and wide-spread vulnerabilities. We will continue to keep an eye on these trends over the coming years and continue to hope that the success or failure of eradication efforts for these vulnerability categories will be a bellwether for progress in application security.

Quarterly Trend for XSS in Government Web Applications

pvalue = 0.215: Statistically, the trend is flat.

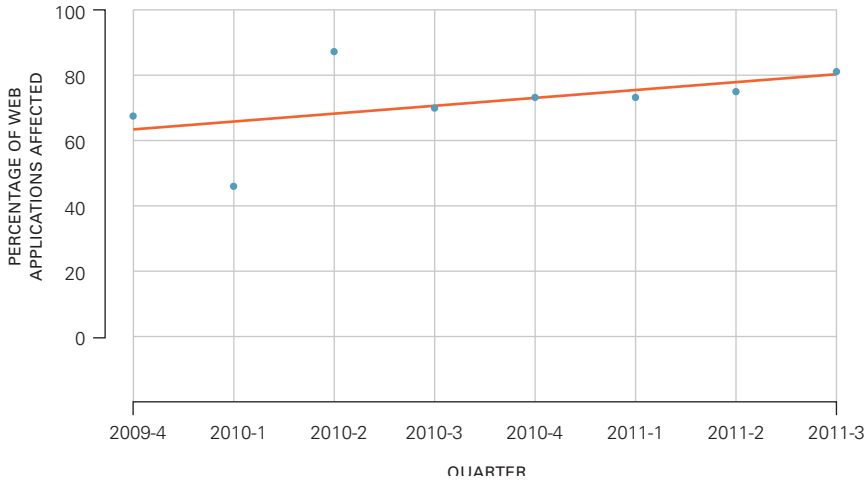

Figure 29: Quarterly Trend for XSS in Government Web Applications

SQL Injection remains flat for Government applications while declining for the overall dataset. Concerning trend given high percentage of reported incidents caused due to SQL Injection.

Quarterly Trend for SQL Injection in Government Web Applications

pvalue = 0.343: Statistically, the trend is flat.

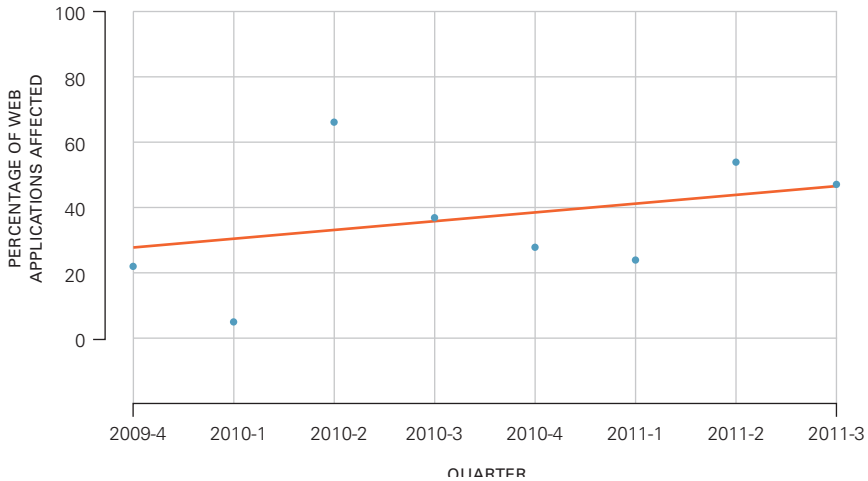

Figure 30: Quarterly Trend for SQL Injection in Government Web Applications

Quarterly Trend for Information Leakage in Government Web Applications

pvalue = 0.198: Statistically, the trend is flat.

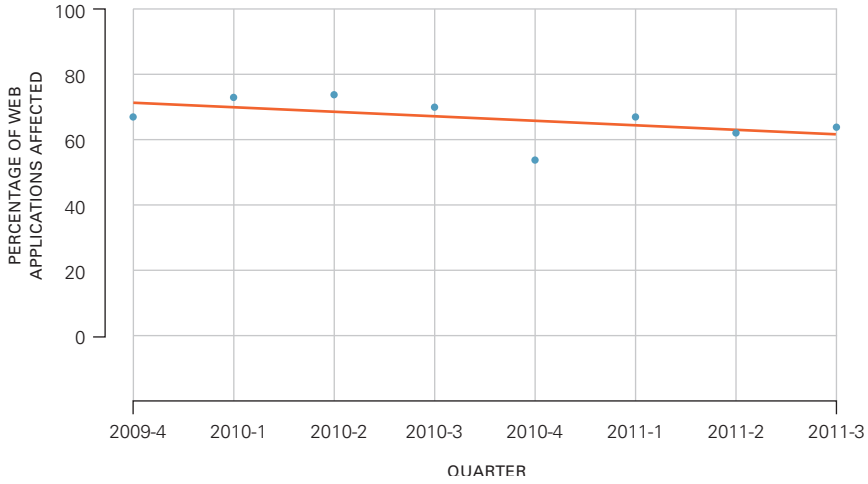

Figure 31: Quarterly Trend for Information Leakage in Government Web Applications

Time to Acceptable Security Quality by Government Applications

Figure 32 depicts remediation behavior for the Government sector for internally developed and commercial supplier types (the other supplier types were a very small segment of the Government dataset so we have not mapped them). Note that the set of applications used to create this figure includes only Government applications that ultimately achieved acceptable security quality according to the Veracode level policy. For the supplier types shown, remediation performance is quite similar. Over 80% of applications that achieved acceptable quality attained it in less than one week. Attaining acceptable quality took over four weeks for 18%% of this dataset. Government applications share very similar remediation performance with all applications, namely, that most applications that achieve acceptability do so quickly.

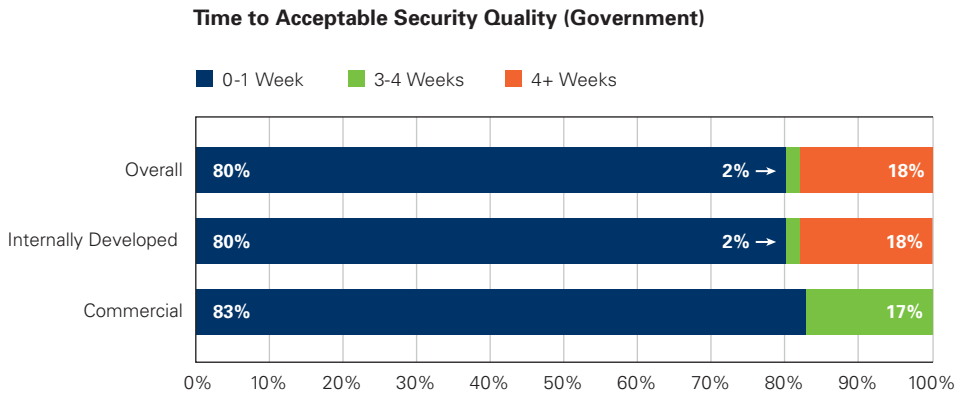

Figure 32: Time to Acceptable Security Quality (Government)

Figure 33 shows Government remediation performance as compared to other industry segments in our dataset. The Government sector's remediation performance, as measured by time to acceptance for applications that eventually achieve acceptance, is comparable to that for the Financial and Software sectors. 80% of Government applications reaching acceptability did so in one week as compared to 71% for Software and 76% for Finance.

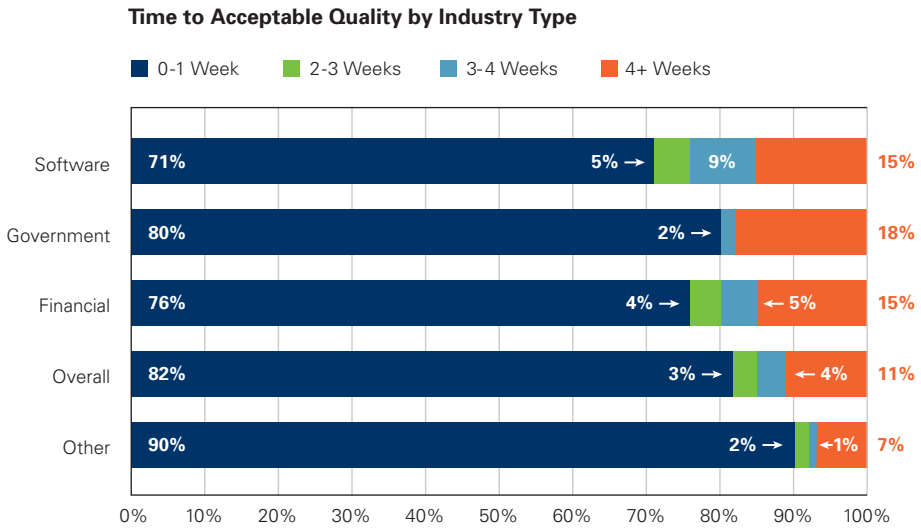

Figure 33: Time to Acceptable Quality by Industry Type

Developer Training and Education

In Volume 3 of this report, Veracode began looking at the state of developer training and education in application security to understand the construction side of the application security challenge. The common hypothesis has been that developer training is an important component of application security programs, since it provides developers with guidance aimed at preventing the introduction of security flaws into applications in the first place. As before, this report considers the performance of developers on graded assessments as a proxy for the state of their application security knowledge (see Figure 34, page 46). The report also examines whether developers who do well on eLearning courses produce applications with higher security quality scores. The Veracode eLearning service offers multiple computer based training and assessment offerings aimed at different points in the developer's learning process:

- Assessments: These are graded tests typically taken at the beginning of a developer's eLearning experience and are intended to measure their baseline knowledge of application security fundamentals in a language-neutral context. Veracode offers a standard assessment (the Veracode Application Security Fundamentals Assessment) for this purpose; though companies may substitute their own assessments, only performance on the standard assessment is considered in this volume.
- Courses: These consist of 30 to 120 minute Computer Based Training sessions that focus on a particular topic, and may be taken ad lib or as part of a structured curriculum. Student performance on some courses is measured via a grade based on an accompanying exam; for other courses, only the completion of the course is recorded.
- Exams: These are graded tests typically (but not necessarily) taken after completing an accompanying course. Exams measure mastery of a more specific set of application security knowledge. This report considers three exams: Secure .NET Coding, Secure Java Coding, and Introduction to Cryptography.

More than half of all developers get a grade of C or lower on the application security fundamentals assessments. Top performers (those achieving grade of A) declined from 31% in Volume 3 to 27% in Volume 4.

As before, 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, and F less than 60.

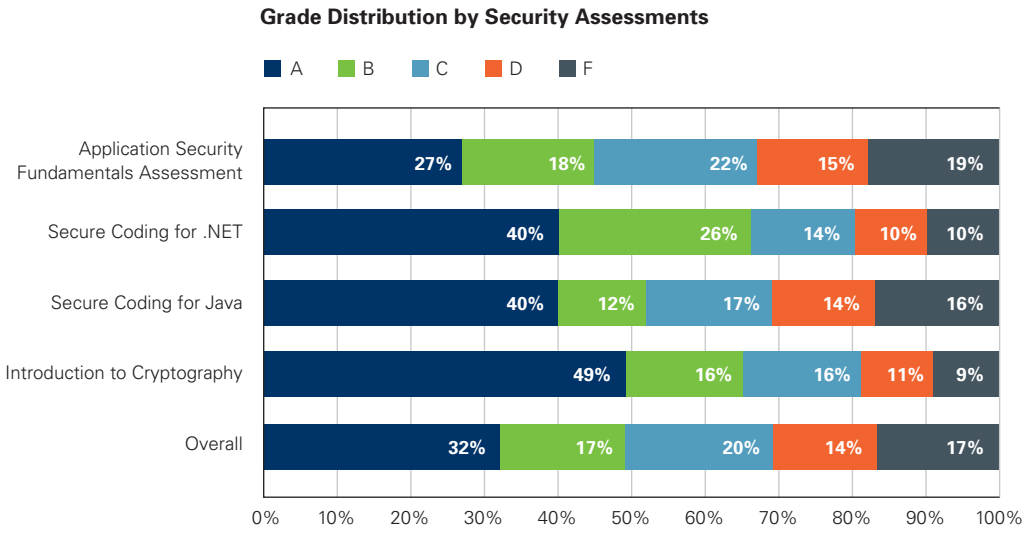

Figure 34: Grade Distribution by Security Assessments

Measurement of initial security knowledge: As in Volume 3, we looked at performance on the Veracode Application Security Fundamentals Assessment as a proxy measurement for initial security knowledge. Compared to Volume 3, student performance on this exam declined, with 27% achieving an A compared to 31% in Volume 3, and 34% achieving a D or F (up from 31%). Overall this reinforces our findings from Volume 3: while some developers measured had a good grasp of application security fundamentals, a larger population has a poor understanding of basic application security principles prior to completing any coursework.

Measurement of developer knowledge: Figure 34 also shows developer performance on three topic-specific exams, taken in combination with an accompanying eLearning course. Here the results were the same as or better than Volume 3. Performance on the Java exam was virtually identical: the same percentage (40%) achieved an A and 30% received a D or F. For the .NET exam, 40% received an A, a slight increase from Volume 3 (35%), while about the same portion failed (20%). Finally, as in Volume 3, students taking the Introduction to Cryptography exam had the best overall scores (49% achieved an A, 20% a D or F). The higher portion of students receiving an A on the topic specific exams can be attributed to the effectiveness of the courses, but the fact that a fifth or more students received a failing grade on each exam points out a continuing education challenge.

Figure 35 represents the percentage of students completing the different assessments discussed in Figure 34. A slightly larger percentage took the Application Security Fundamentals Assessment compared to Volume 3 (62%, up from 48%), a lower percentage took Secure Coding for .NET (6% down from 16%) and comparable with percentages completing the Java and Introduction to Cryptography exams.

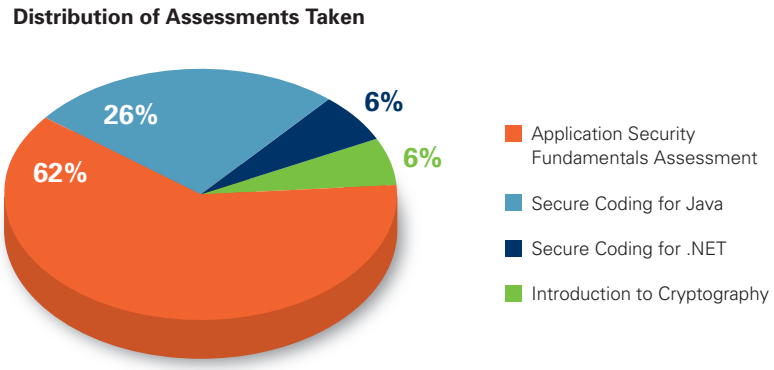

Figure 35: Distribution of Assessments Taken

Impact of Security Education on Application Security Quality Score

The reason that developer education plays an important role in an application security program is the hypothesis that an educated developer produces fewer security flaws, thus lowering the overall application security risk of the enterprise at a lower cost than remediating vulnerabilities later in the SDLC. Figure 36 (page 48) explores that hypothesis by exploring the relationship between average performance on developer security assessments and average software quality score on the Veracode Application Security Fundamentals Assessment.

For the purposes of Figure 36, the dataset shows average software quality score only for those applications that were not assessed via a formal third party review, and only application builds classified as under development, in alpha testing, or in beta testing. The intention of these restrictions was to focus on newer applications (under active development or in pre-production testing) that could reasonably have been written with modern development practices and have benefited from developer security education, and to eliminate applications where the enterprise developer (being measured by the assessment) was not the primary author.

The average software quality score was plotted against the average performance on the Veracode Application Security Fundamentals assessment. This assessment was chosen because it represented the broadest sample of the developer population and did not test language specific knowledge.

The results in Figure 36 do appear to show a relationship between average organizational application security knowledge, as measured by the Veracode ASF assessment, and application software quality as measured by the average application software quality score. However, the relationship should not be read as a strict correlation, and indeed should not be expected to be. There are multiple factors that complicate the attempt to establish the relationship between developer security knowledge and software quality, including but not limited to the likely presence of embedded vulnerabilities in third party or legacy code that the developer being measured did not write, the lack of information about which parts of an application and any corresponding flaws that came from any one developer being tested, and so on.

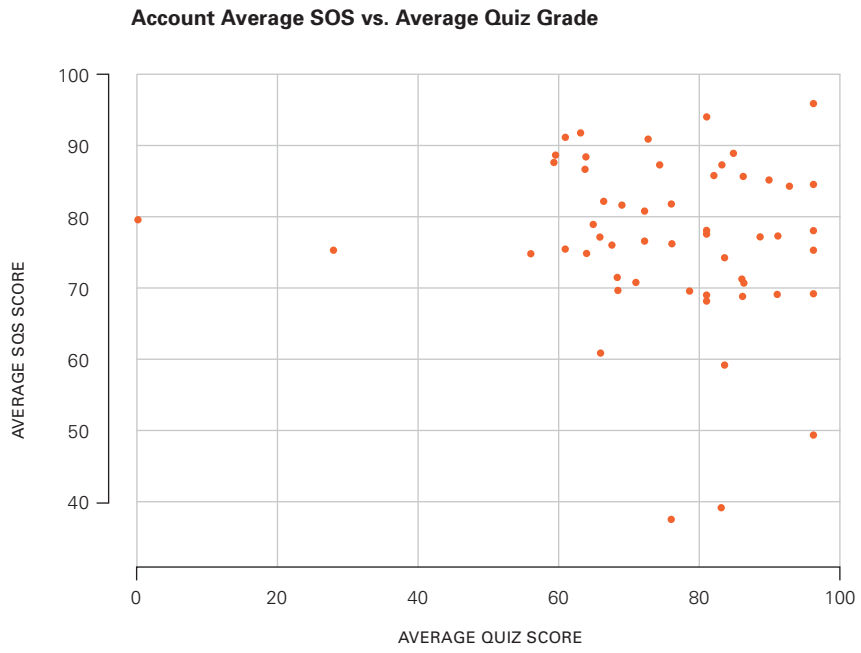

Figure 36: Account Average SOS vs. Average Quiz Grade

However, despite all these caveats, the data does suggest that it may be reasonable to use an assessment like the Veracode Application Security Fundamentals Assessment to suggest likely software quality; even if the precise relationship is unclear, the presence of even an indirect relationship is encouraging. Future volumes of the State of Software Security will attempt to explore this point further.

Application Threat Space Trends

The security world has been shaken up since our last State of Software Security report earlier in 2011. There has been steady reporting of major attacks from three distinct groups of threat actors: criminal, espionage, and hacktivist. The number and severity of breaches has increased but it is unclear whether this is due to an actual increase in breaches or merely an increase in reporting. Reporting is certainly up as organizations feel compelled to report serious breaches in the interest of protecting their customers, even if not required by law. Hacktivists self-report breaches as part of a strategy to embarrass their victims.

Another trend is that attacks against one party are being used as a building block to stage attacks on other valuable targets. For example, the RSA breach was leveraged to gather information that enabled attacks against RSA's customers. Only weeks after the RSA attack came to light Lockheed Martin was compromised, allegedly using information stolen from RSA. Gawker, Silverpop, and Epsilon were also attacked and while the stolen data was limited to usernames, email addresses, and passwords, this data was likely used in follow-on attacks such as spear phishing and targeting popular webmail and social networking sites where users commonly reuse passwords. Adversaries are gaining sophistication and thinking strategically in attack campaigns instead of just conducting a string of one off tactical attacks.

The increased breach reporting has led to greater insight on attacker techniques. While most criminals are likely to use phishing to compromise many machines and gather passwords, some criminals and certainly the cyber espionage actors are moving to spear phishing. This is another sign of increased sophistication as spear phishing typically requires a team approach where gathering information about targeted individuals is just as important as expertise in malware and exfiltration.

Exploiting Web Application Vulnerabilities

A prominent attacker technique on the rise is exploiting vulnerabilities in web applications. In the past, these vulnerabilities were typically used to steal data, but their utility for attackers is broadening. Web application vulnerabilities are being exploited to:

- Steal PII and financial data
- Modify victim web sites to serve up malware for secondary attacks against web site visitors or simply to embarrass the victim
- Steal names and email addresses to be used in spear phishing campaigns
- Steal email addresses and passwords to take over high value accounts if there is password re-use
- Breach organization perimeters where web applications are behind the organization firewall or have access to internal resources

We can clearly say "this isn't your father's SQL injection attack" anymore. The same web vulnerabilities that have been somewhat problematic in the past are being exploited in ways that significantly impact both the vulnerable organization and all the downstream victims of secondary attacks.

This pattern of using web application vulnerabilities far beyond just stealing data means that the population of targeted web applications has also grown. Practically every web application, business critical or not, has exposure to financial data, customer information, or intellectual property. Every web application is also an extension of the corporate brand and must be protected accordingly.

Seemingly low risk web applications can still be valuable targets due to shared resources on an organizations network such as databases and file storage. A vulnerability in a low risk application combined with shared resources that aren't properly isolated can lead to a critical breach. This was the attack vector used in the Heartland Payment Systems breach and the more recent Night Dragon attacks. Even a long forgotten marketing web site for something like a product launch can be leveraged by attackers to cause serious harm.

With attackers casting a wider net, organizations are increasingly strained. While they were once concerned with testing and securing a handful or perhaps dozens of web applications, they are now confronted with the challenge of testing and securing hundreds or even thousands of applications.

Web Vulnerabilities Attacked

The target population of attacked web sites is increasing dramatically. On the other hand, the vulnerability categories being exploited at scale is still quite limited. Analysis of tweets, chat logs, and pastebin postings shows that hacktivists have focused primarily on three categories: [10]

- Remote file includes
- SQL injection
- Cross-site Scripting

Another collection of data from real-world incidents is the Web Hacking Incidents Database (WHID), a project of the Web Application Security Consortium (WASC). This database covers all types of incidents, not merely those conducted by hacktivists. The latest WHID statistics for top hacking methods[11] list the following top attack techniques:

Top Hacking Methods

Attack Method	Percentage
Unknown	22.9
SQL Injection	20.2
Denial of Service	11.8
Cross-site Scripting	9.4
Brute Force	3.8
Predictable Resource Location	3.7
Information Disclosure	3.2
Credentials/Session Prediction	2.2
Stolen Credentials	2.1
Cross-site Request Forgery	2.0

Table 8: Top Hacking Methods

[10] blog.imperva.com/2011/06/analyzing-the-lulzsec-attacks-.html

[11] projects.webappsec.org/w/page/13246995/Web-Hacking-Incident-Database

The top 10 attack methods account for 80.4% of the web application compromises in the WHID. 22.9% is categorized as unknown but it is likely that the majority of those attacks used one of the popular techniques and there just wasn't enough information to conclusively determine the attack method.

Knowing that there is a limited set of vulnerability categories that are exploited in most attacks and the likely targets of those attacks allows for a risk based approach to application security testing and subsequent remediation. An application security program that covers all web applications for only the top vulnerability categories can reduce attack surface in a cost-effective manner. This is where most organizations should start their applications security program given the current threat space.

Mobile

Mobile applications are being produced at a dizzying pace that is beginning to rival the production of web applications. The majority of these mobile applications are personal productivity, entertainment, or gaming applications. However, mobile ecosystems are maturing and beginning to resemble other mature platforms such as the desktop OS or the web. Over the last year we are starting to see many financial, healthcare, and other enterprise class applications. Directors in board rooms are now sharing sensitive corporate documents using iPad applications like NASDAQ's Directors Desk. Doctors are controlling medical equipment such as anesthesia carts with iPad applications. Mobile operating systems do have a smaller attack surface and improved security features over desktop and web applications, but these applications can still fall prey to vulnerabilities that affect the traditional security areas of confidentiality, integrity, and availability.

Veracode introduced support for the popular Android and iOS mobile platforms this year due to requests from customers. Rather than waiting for the mobile threat space to heat up, many finance and healthcare enterprises are already building application security into their mobile application SDLCs and are holding their mobile application developers accountable for security. We have published the Top 10 Mobile Application Risks[12] to educate developers and software acquirers of the biggest risks.

Currently, the most prominent mobile threat is malicious applications. In 2011, several malicious applications including DroidDream[13] have been removed from the Android Marketplace after being downloaded by hundreds of thousands of users. In November 2011 the Apple App Store removed a proof of concept application that demonstrated how malicious, unsigned code could have been planted in applications downloaded from the Apple App store, circumventing their code signing protections.[14]

[12] www.veracode.com/blog/2010/12/mobile-app-top-10-list

[13] www.readwriteweb.com/archives/30000_to_120000_android_users_affected_by_new_variant_of_droid_dream_malware.php

[14] techland.time.com/2011/11/14/the-consequences-of-apples-walled-garden

Jailbreaking

An important contributor to the mobile application threat space is the jailbreaking community. Jailbreaking relies on any number of privilege escalation vulnerabilities that allow the jailbreaker to run code at the root or system level in order to modify the mobile operating system. The jailbreaker has one goal in mind: to modify the OS so it will run unsigned code not sanctioned by the mobile OS provider. The jailbreaker seeks to discover privilege escalation vulnerabilities, publish tools that use them, and not inform the OS vendor lest the tool be rendered useless. This is an advantageous scenario for malware authors because they can extract working privilege escalation exploit code from publicly available jailbreak tools. This scenario is not theoretical; the DroidDream malware contained exploit code that had been published by a jailbreaker. Any malware or code execution vulnerability is game over in this environment.

More Attackers and More Applications Define Application Security Programs

The overall trend in the application security threat space is an increased variety of threat actors and a wider organizational attack surface, with most if not all web applications being targeted. Cyber espionage and hacktivism seem to be here to stay, and the volume of web and mobile applications will continue to rise. A risk based application security approach is the most cost effective way to meet these challenges. Look to the threat space for the types of applications under attack and the vulnerabilities targeted. Then design an application security program that can eliminate the types of vulnerabilities attacked from the applications likely to be targeted.

This risk based approach will require scaling existing application security programs to handle many more applications than most organizations are already covering. This will likely call for more and better automation. The good news from the threat space is that a reasonably small number of vulnerability categories are being actively exploited by attackers. So while more applications need to be inspected and remediated, focusing on specific categories can yield significant security improvements. Application security is a journey that should be guided by the threat landscape and revisited as the landscape evolves.

Addendum

Veracode Level Policy Overview

Veracode 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 Business Criticality

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

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

Business Criticality Descriptions

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

Very High (BC5)

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 (BC4)

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 (BC3)

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 (BC2)

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 (BC1)

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.

About Acceptable Software Quality

In this report, Veracode presents four means by which an application may be judged to be acceptable: CWE/SANS Top 25, OWASP Top 10, Veracode Levels, and enterprise policy. When applications are judged acceptable against CWE/SANS Top 25 or OWASP Top 10, the flaws in the application are compared to the flaw categories represented by the respective standards. (Veracode uses the CWE mappings to determine whether a flaw belongs in the OWASP Top 10 or the CWE/SANS Top 25.)

Veracode Levels is Veracode's new standard for evaluating the software quality of an application, which combines consideration of the security quality score of an application, the severities of the flaws contained therein, and the types of testing performed. Applications are given a risk-adjusted Veracode Level target based on the business criticality of an application; an application with a low Business Criticality need not receive the highest Veracode Level score. The criteria for Veracode Levels and the target levels for each Business Criticality are defined below.

Criteria for Veracode Levels and Target Levels for Business Criticality

Business Criticality	Target Veracode Level	Flaw Severities Not Allowed	Testing Required	Minimum Security Quality Score
BC5	VL5	Very High, High, Medium	Static and Manual	90
BC4	VL4	Very High, High, Medium	Static	80
BC3	VL3	Very High, High	Static	70
BC2	VL2	Very High	Static or Dynamic or Manual	60
BC1	VL1	Not Applicable	Static or Dynamic or Manual	Not Applicable

Table 10: Criteria for Veracode Levels and Target Levels for Business Criticality

The Veracode Platform allows customers to use one of these measures for acceptable quality or to define their own policy, which constitutes the definition of an acceptable level of risk for the application.

About the Dataset

The data represents 9,910 application builds submitted for analysis by large and small companies, commercial software providers, open source projects, and software outsourcers. In most analyses, an application was counted only once even if it was submitted multiple times as vulnerabilities were remediated and new versions uploaded. The report contains findings about applications that were subjected to static, 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 Level, enterprise policy status).

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.

VERACODE

Veracode, Inc.
4 Van de Graaff Drive
Burlington, MA 01803

Tel +1.781.425.6040
Fax +1.781.425.6039

www.veracode.com

SSSR/US/1211

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 that help identify and eradicate numerous vulnerabilities by leveraging best-in-class technologies from vulnerability scanning to penetration testing and static code analysis. 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. Veracode works with customers in more than 80 countries worldwide including Global 2000 brands such as Barclays PLC and Computershare as well as the California Public Employees' Retirement System (CalPERS) and the Federal Aviation Administration (FAA). For more information, visit www.veracode.com, low on Twitter: @Veracode or read the ZeroDay Labs blog.