Cognizant 20-20 Insights

# Application Vulnerability: Trend Analysis and Correlation of Coding Patterns Across Industries

Using our latest assessment, security architects and developers can determine which industries — as well as areas of source code and applications — are most vulnerable to attack, and mitigate the impact.

## Executive Summary

Attacks on Web applications threaten nearly every organization with an online presence. Based on our experience, these unwelcome assaults cost companies millions of dollars and can cause serious damage in terms of brand integrity and customer turnover. Our Enterprise Risk and Security Solutions (ERSS) assessment team recently evaluated the state of Web application vulnerability using automated vulnerability scanners and manual tests to analyze the state of security across nine industries.

This white paper presents results identified during 2012 and 2013. It focuses on the general application functionalities and the corresponding parameters that were developed, but failed, to secure code across verticals. The paper also details suitable recommendations for mitigating security vulnerabilities that arise within these scenarios.

## Vulnerability Analysis

The security posture of each vertical analyzed during our 2013 assessment can be best understood by examining the concentration of vulnerability across these industries (see Figure 1, page 2). Vulnerabilities pertain to severity levels – high, medium and low. Applications within the insurance vertical comprise the highest percentage in total vulnerabilities across the verticals. These applications also contain the highest number of security coding flaws or static application vulnerabilities. Banking and Financial Services (BFS) and Information, Media and Entertainment (IME) applications have nearly the same vulnerability levels, with IME applications being the most susceptible – showing the highest number of dynamic application vulnerabilities compared with other verticals.

## Vulnerability Trends in 2012 and 2013

The security posture of Web-based applications is continuously changing, primarily due to the rise of new hacking methods, the spreading awareness among developers and regulatory compliance, for example.[1] Figure 2 compares application vulnerability distribution across various verticals, based on the findings of SAST and DAST assessments conducted in 2012 and 2013. The number of applications tested is also shown.

In 2012, nearly 76% of the vulnerabilities we identified were found in healthcare applications,

## Vulnerability Concentration Across Verticals

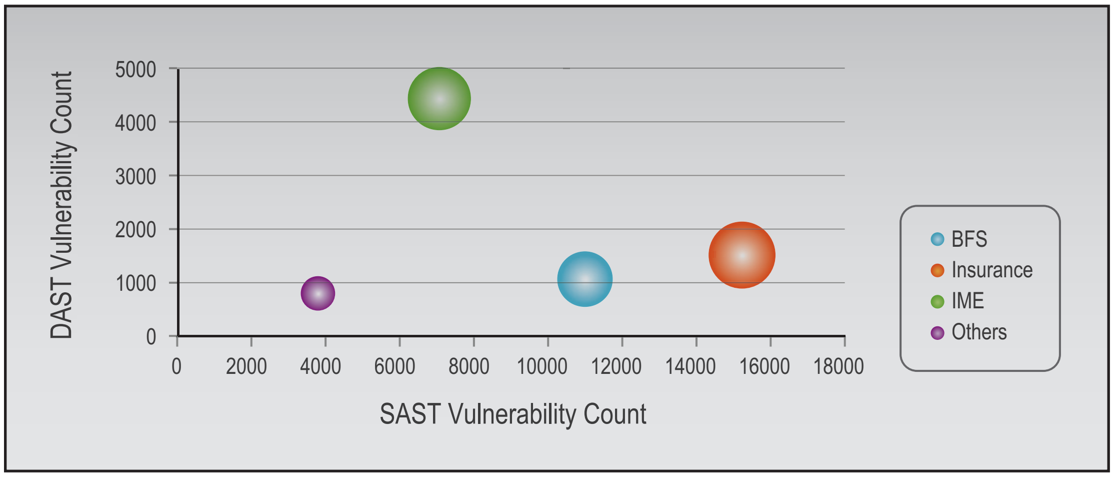


Figure 1

14% were discovered in insurance industry applications, and about 3% were identified in BFS applications. Fewer vulnerabilities were seen in retail, IME and other domains such as travel and hospitality and consumer goods. In 2013, nearly 37% of the vulnerabilities were detected in insurance industry applications, 27% were found in BFS, 26% in IME and 8% in retail.

The following sections describe our industry-based vulnerability analysis. The study used automated vulnerability scanners and manual tests, and employed SAST- and DAST-specific interpretation of industry trends to zero in on the exact application threats and their causes.

## Static Application Security Testing

In static application security testing (SAST), the application code is examined for flaws that can lead to security threats. SAST uses tool-based scanning, as well as manual reviews. Tool-based scanning involves tests generated by pre-defined security rules. Manual review entails validating the tool output and identifying additional security flaws using manual expertise.

### Automated Tool for Vulnerability Detection

Tools for automated security testing produce results with false-positives (identified as application vulnerabilities by the tool, but not actually

## Vulnerability Concentration Across Verticals

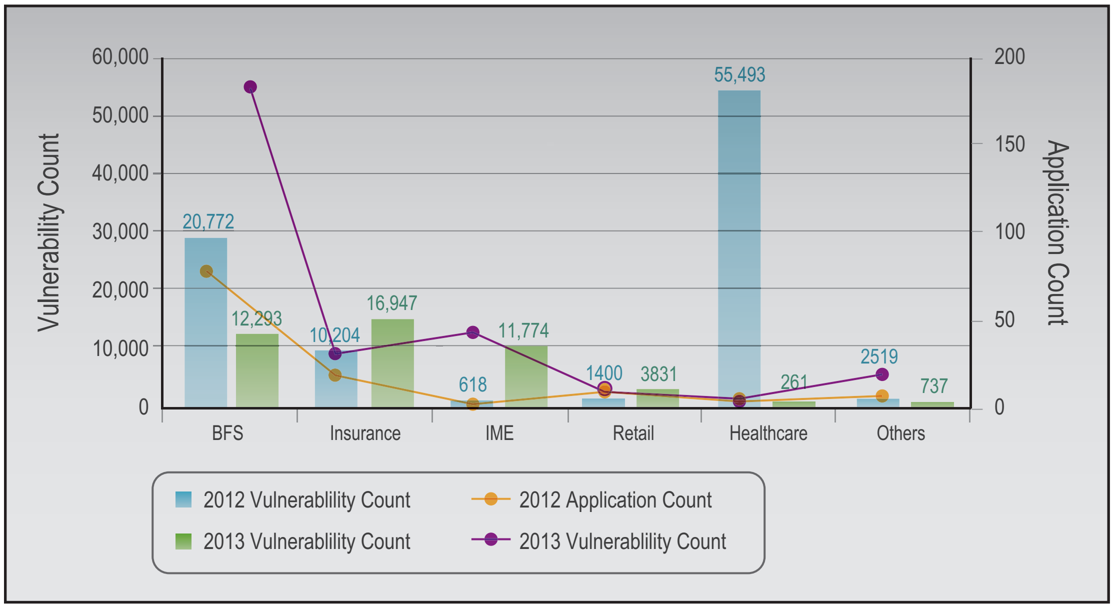


Figure 2

## Automatic SAST Tool for Vulnerability Detection

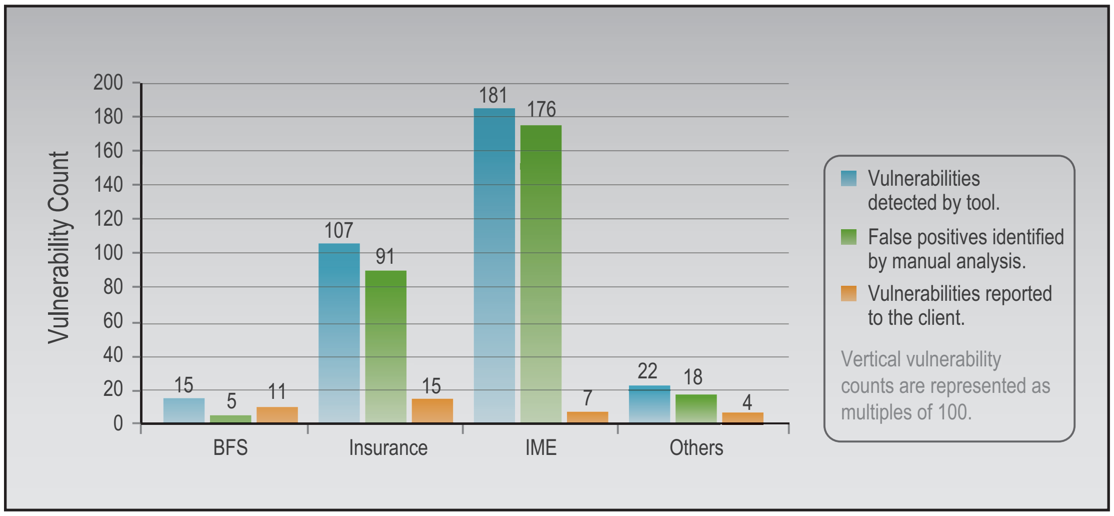


Figure 3

vulnerabilities) and false negatives (existing vulnerabilities that were missed by the tool).

Manual analysis techniques are employed to eliminate false-positives and identify false-negatives. Figure 3 above illustrates the summary of the number of coding flaws identified by the automated security code scanning tools, false-positives identified by manual analysis and the actual vulnerabilities reported to the client's point of contact for the applications of different verticals. The major verticals assessed here were BFS, Insurance and IME. Other verticals, including retail, healthcare, T&H and mobility, were grouped into one category.

For this analysis, we employed a number of commercial scanners, open source tools and freeware. As shown by the data, automated security scanners have huge false-positive rates. For example, in the insurance and IME verticals, more than 90% of reported issues are false-positives. In general, applications in IME verticals rely more on Web 2.0 components, Flash and Action scripts, which can increase their complexity. Automated tools are very limited when it comes to understanding the business logic and functional flow of the applications, due to the high false-positive counts found in this vertical. This makes the intervention of manual security expertise essential — not only for removing false-positives, but also for uncovering vulnerabilities in the application that automated tools fail to capture.

## Vulnerability Trends in Verticals

Statistical information about the vulnerabilities pertaining to SAST with respect to different verticals and vulnerability categories is depicted in Figure 4 (next page). Security standards such as OWASP[2], WASC[3] and CWE[4]/SANS[5] were used to classify these vulnerabilities.

The various categories of secure coding flaws for different verticals are listed in Figure 5 (next page). The most prevalent of these falls under the "Best Practices Violation" category, due to the lack of awareness among developers concerning adherence to secure coding standards. Common poor coding practices include null pointer dereference, missing checks against null, using weak XML schema, data in hidden fields and failure to remove debug code, comments and other sensitive leftover code. "Information Leakage, Error Handling and Input Validation" flaws are also rampant due to improper handling of application input and output, which form the major entry points for application attacks. Of the total issues identified, insurance industry applications were found to contain the highest number of security coding flaws. In fact, 91% of coding flaws were found in "Best Practices Violation," followed by 5% in "Input Validation" and 3% in "Information Leakage and Error Handling" categories.

## Dynamic Application Security Testing

Dynamic application security testing (DAST) or "black-box" testing evaluates applications

## SAST Vulnerability Distribution

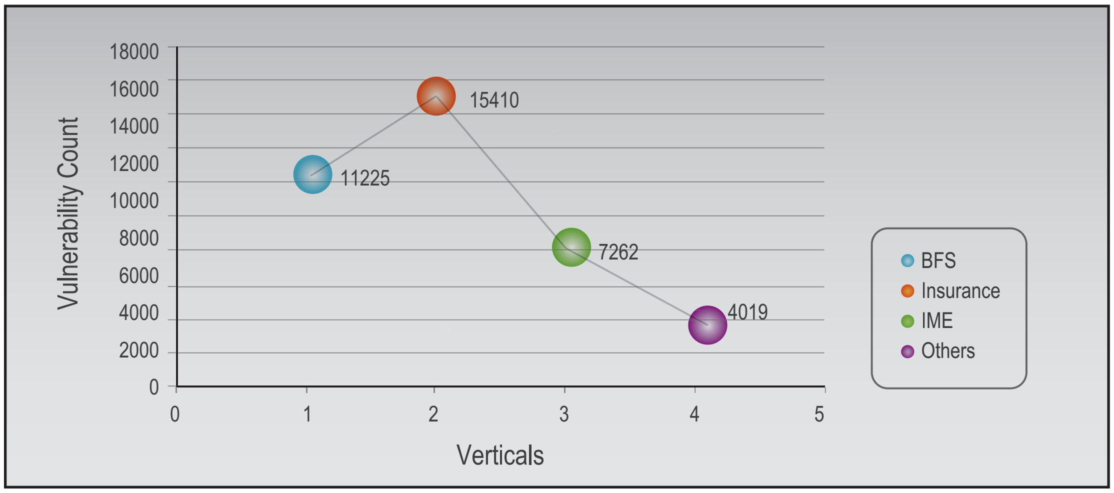


Figure 4

during their execution at runtime. This is useful in determining the risks the application faces in a production environment. Our ERSS team employs automated scanning tools and manual testing techniques to dynamically test an application. The following sections of this white paper elucidate DAST vulnerability detection using automated tools and industry-based DAST vulnerability trends.

### Automated Tool Vulnerability Detection

Dynamic testing is performed using industry-standard automated scanners. The performance of each scanner typically depends on the security rule sets defined for these tools. Cognizant's ERSS group performs intensive manual testing, which helps assure comprehensive coverage. Some of the manual tests include detecting business logic bypass issues and session-related problems, such as session hijacking, session fixation and session replay, as well as authentication issues like insufficient logout mechanism, improper cache management, and security misconfiguration issues such as SSL renegotiation, click jacking and other such vulnerabilities. Figure 6 (next page) summarizes the number of application vulnerabilities identified by various automated dynamic security testing tools (commercial, open

## Vulnerability Count Based on SAST

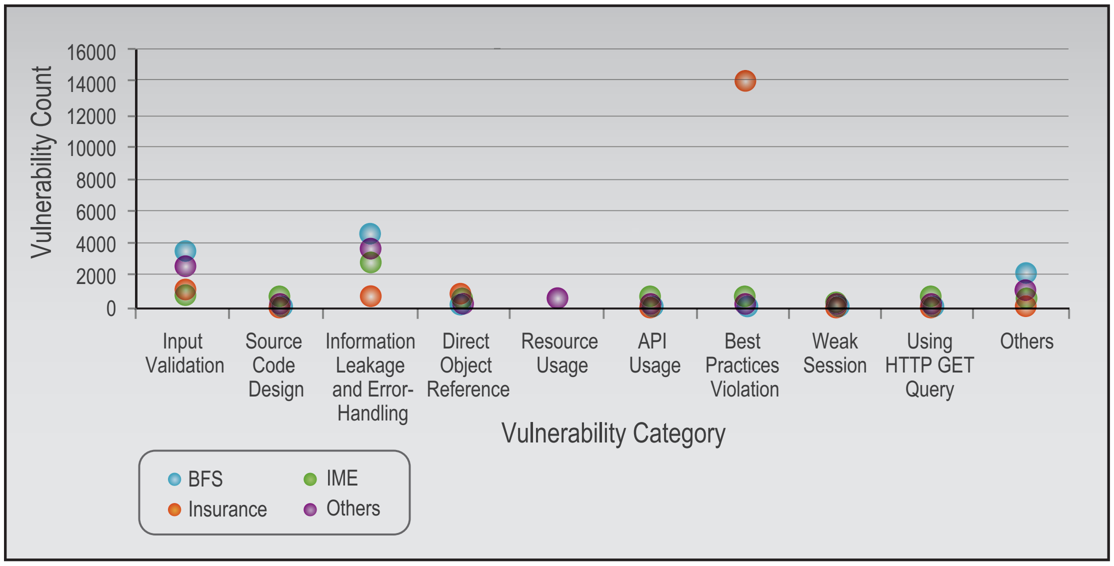


Figure 5

## Automatic DAST Tool Vulnerability Detection

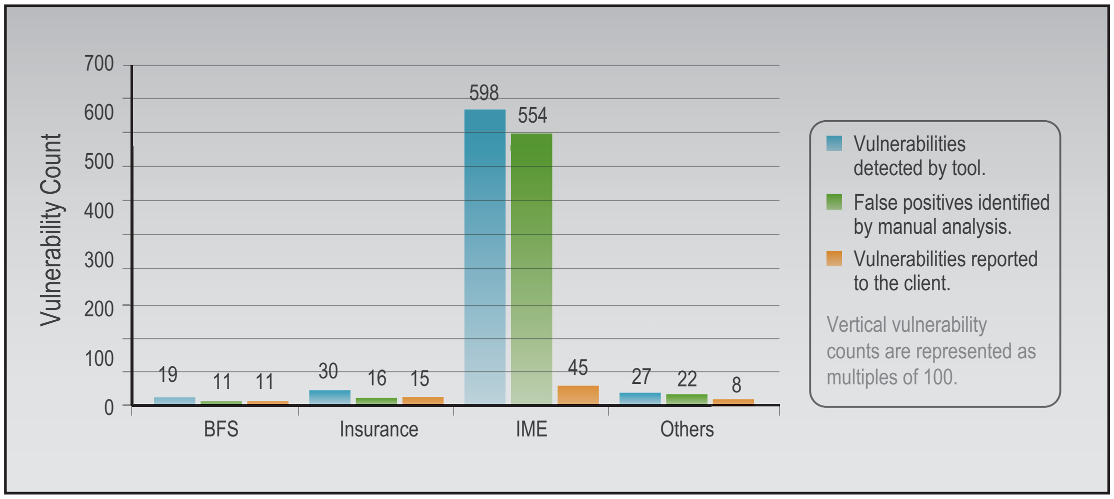


Figure 6

source and freeware), false-positives identified by manual analysis, and the actual vulnerabilities of different verticals reported to the client.

### Vertical Vulnerabilities

The number of dynamic application vulnerabilities is showcased across verticals (see Figure 7). Most vulnerabilities were found in the IME vertical, with the highest count being in the "Insecure Direct Object Reference" category, followed by "Injection." Next in line was the insurance vertical, with the highest count in "Security Misconfiguration," followed by "Insufficient Transport Layer Protection."

As Figure 8 (next page) shows, the most dominant vulnerability was in the "Insecure Direct Object Reference" category. When a developer exposes a reference to an internal object to the user, this type of vulnerability occurs. A large number of vulnerabilities were also found in "Injection" and "Cross-Site Scripting," denoting that developers still show their trust in user input by failing to perform sufficient input validation and output encoding, and using secure defaults. "Security Misconfiguration" and "Insufficient Transport Layer Protection" were also very prevalent. This could result from testing environments that do not mirror the actual production environment, have weak server configurations, or have no or poor SSL configurations. Robust configurations are essential for maintaining high security for a live site compared with a test site.

## DAST Vulnerability Distribution

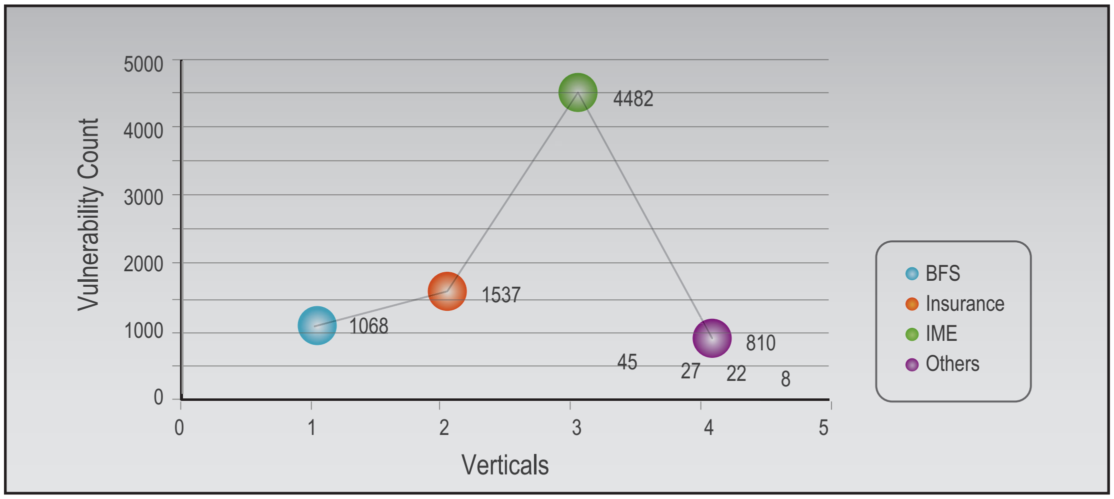


Figure 7

## Vulnerability Count Based on DAST Category

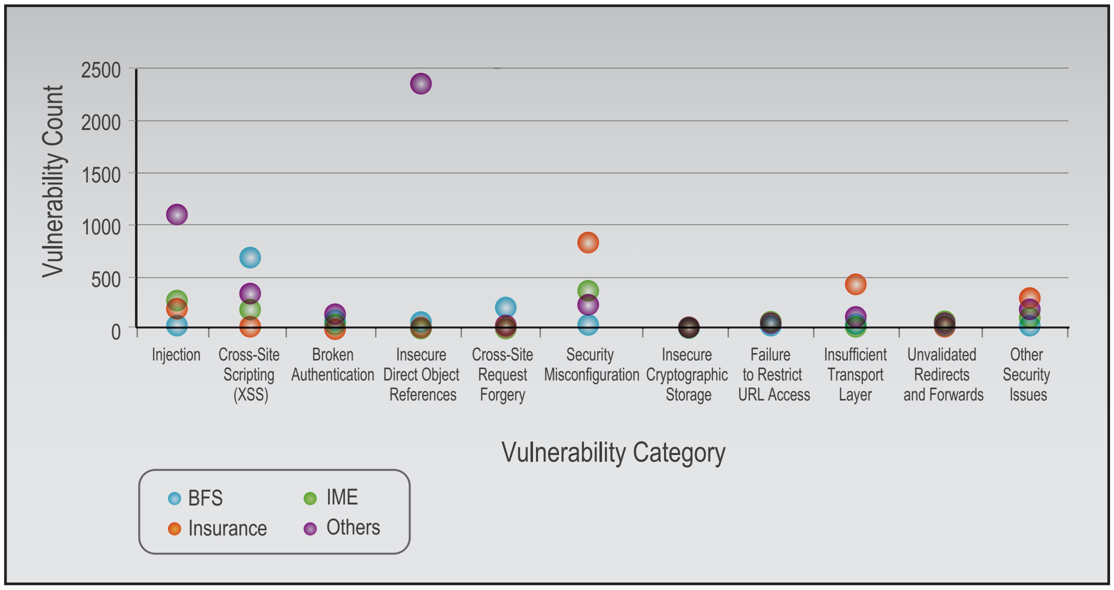


Figure 8

The overall vulnerability count in the DAST and SAST findings was highest in IME applications. The total count is comparable to those of insurance and BFS. In fact, nearly 60% of the applications tested belong to BFS, but only about 25% of the total vulnerabilities reported are present in those applications. Furthermore, these applications have fewer critical vulnerabilities related to issues such as injection. Hence, it can be inferred from the statistics that BFS applications are relatively more secure. This can be attributed to the fact that the banking and finance sector deals with highly sensitive data, for which security is paramount. While confidentiality and integrity of financial data are critical, the third parameter of the security triad – availability – is equally essential for this growing industry. Security awareness is progressively increasing within the developer community in BFS because of these requirements, thereby leading developers to emphasize application security. And because BFS applications are also subject to compliance mandates, security requirements are taken care of during development. This helps to keep these applications even more secure.

## Security Trends in Application Development

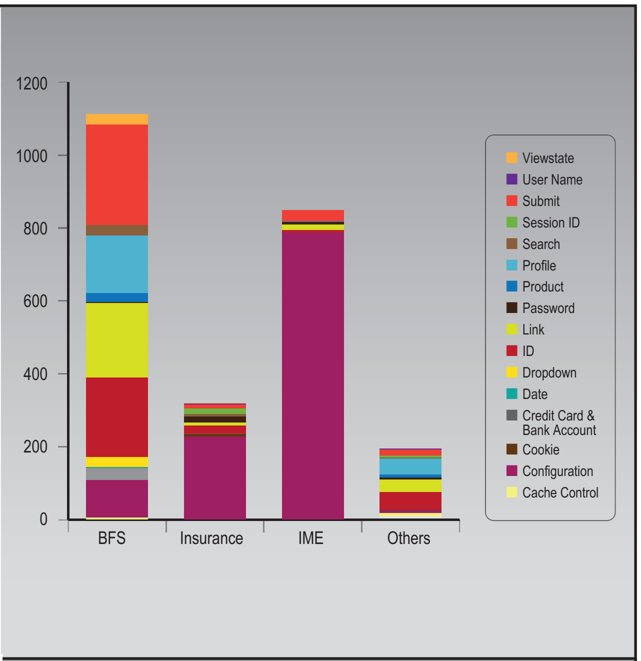
Security threats in an application can be drilled down to the particular functionality that is affected by the security flaw and the parameter that provides an entry point to the attack vector. Typical application functionality could be anything from a login/logout function to a payment function. It can further be zeroed down on the query parameters, form parameters, cookies and page parameters that are created by the developers to accomplish the respective functionality. Figure 9 (next page) shows the distribution of vulnerable parameters and functions across verticals.

The most commonly affected parameters are the configuration parameters, which impact the configuration function and make the application susceptible to security misconfiguration issues. This is due to failure to employ platform-specific secure configurations. Developers should also focus on add/modify/submit functions, which are largely vulnerable; submit parameters, for instance, are often targeted by attackers. As a result, applications become prone to cross-site request forgery, clickjacking and malicious content uploads.

These parameters can be safeguarded during development by setting secure attributes and performing safety checks to ensure that the data or file that is submitted conforms to the accepted type, range and business logic. Other parameters that require attention include URL/links, profile parameters and IDs such as user IDs, session IDs and viewstate parameters. Failure to secure these can result in unsafe redirects and forwards, phishing, session hijacking and user impersonation.

## Vulnerable Parameters and Functions

| Parameters Type | BFS | Insurance | IME | Others |
|---|---|---|---|---|
| Cache Control | 5 | 0 | 0 | 2 |
| Configuration | 103 | 227 | 787 | 6 |
| Cookie | 34 | 0 | 0 | 10 |
| Credit Card & Bank Account | 0 | 8 | 0 | 0 |
| Date | 2 | 0 | 0 | 7 |
| Dropdown | 28 | 0 | 0 | 1 |
| ID | 218 | 22 | 7 | 50 |
| Link | 204 | 9 | 16 | 33 |
| Password | 2 | 16 | 6 | 6 |
| Product | 25 | 0 | 2 | 9 |
| Profile | 158 | 0 | 0 | 43 |
| Search | 29 | 8 | 0 | 3 |
| Session ID | 0 | 14 | 0 | 6 |
| Submit | 275 | 11 | 31 | 16 |
| User Name | 0 | 3 | 0 | 2 |
| Viewstate | 30 | 1 | 0 | 0 |



| Functionality Type | BFS | Insurance | IME | Others |
|---|---|---|---|---|
| Add/Modify/ Submit | 736 | 18 | 27 | 122 |
| Configuration | 147 | 239 | 787 | 18 |
| File Upload/ Download | 13 | 1 | 23 | 17 |
| Login | 0 | 39 | 8 | 9 |
| Logout | 0 | 0 | 0 | 3 |
| Password | 0 | 0 | 0 | 2 |
| Payment | 170 | 12 | 2 | 5 |
| Redirect | 0 | 0 | 0 | 4 |
| Search | 0 | 8 | 0 | 14 |
| User Roles/ Privilege Escalation | 47 | 2 | 2 | 0 |

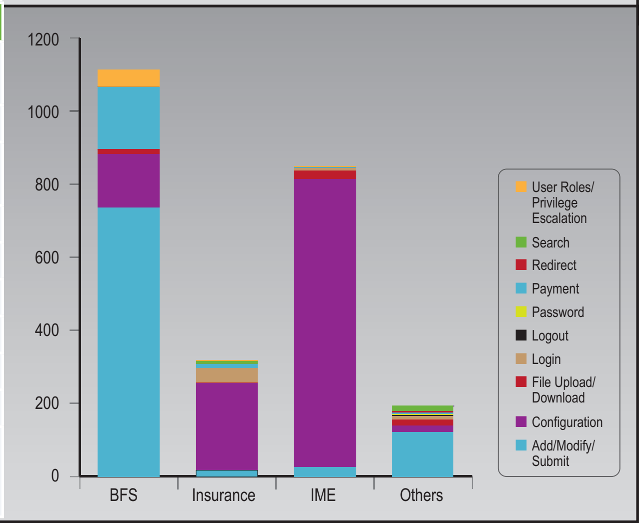


Figure 9

It is the responsibility of the developer to ensure that URL redirects are examined for authorization. Developers should also ensure that ID parameters are generated based on stringent industry-standard protocols, and that session IDs are correctly invalidated — not resused — and regenerated at frequent intervals. These session tokens and other sensitive data must be protected during transit by using proper SSL configuration, and also in cookies in order to prevent cookie theft. Payment parameters are often targeted too, as they can be exploited to execute payment frauds and cybercrimes. Therefore, the duty lies with developers to ascertain that these parameters are handled in a highly secure manner. The focus areas for the developer community should be to incorporate strong validation for input and output parameters, follow secure configurations, set safe attributes for the parameters in general, and preserve the confidentiality of the sensitive data carried by the parameters.

## Looking Ahead

This white paper has presented statistics on application vulnerability trends across several verticals with respect to dynamic and static application security testing. The following recommendations will help developers improve security across numerous parameters:

- Ascertain the validity of input supplied by the user with respect to the data type, range, size and business logic allowed.
- Ensure that session management is robust by using industry-standard session-management and handling mechanisms.
- Protect sensitive data such as IDs, session tokens, user personal data, payment information and the like in transmission and storage.
- Employ secure configurations for all application components.

The developer community in general should closely adhere to security standards and implement secure practices throughout the software development lifecycle to create highly secure applications.

## Analysis Methodology

Figure 10 describes the applications that were studied using dynamic (DAST) and static application security testing (SAST) methodologies. It illustrates a statistical representation of the various applications for which security testing was conducted based on their type, the verticals to which they belong and the technology used.

SAST was performed on 214 applications and DAST on 105 applications. Of the total applications tested, 79% were Web applications, 5% were mobile applications and 16% were other types of applications such as IVR, mainframe, native applications, Web services and CS, for example.

Approximately 60% of Web application assessments were carried out in the BFS vertical, followed by 14% in IME and 10% in the insurance vertical. Mobile application security assessments were performed for the insurance, retail, healthcare, travel and hospitality, IME and manufacturing/logistics verticals.

SAST was carried out on applications that were built using technologies such as Java, Android, .Net, COBOL, Objective C and PHP. Nearly 63% of the applications were developed in Java, Java-based frameworks and Android. .Net projects made up 13% of the applications and COBOL comprised 21%.

SAST was performed on the codebase of Web, native and mobile applications and Web services, for example. Cognizant, as well as third party-developed code, was taken into account for this analysis. SAST for code developed over several frameworks – including mobile/Web frameworks such as Titanium Appecelerator and e-commerce frameworks such as ATG – were also considered.

## Application Distribution by Type, Vertical and Technology

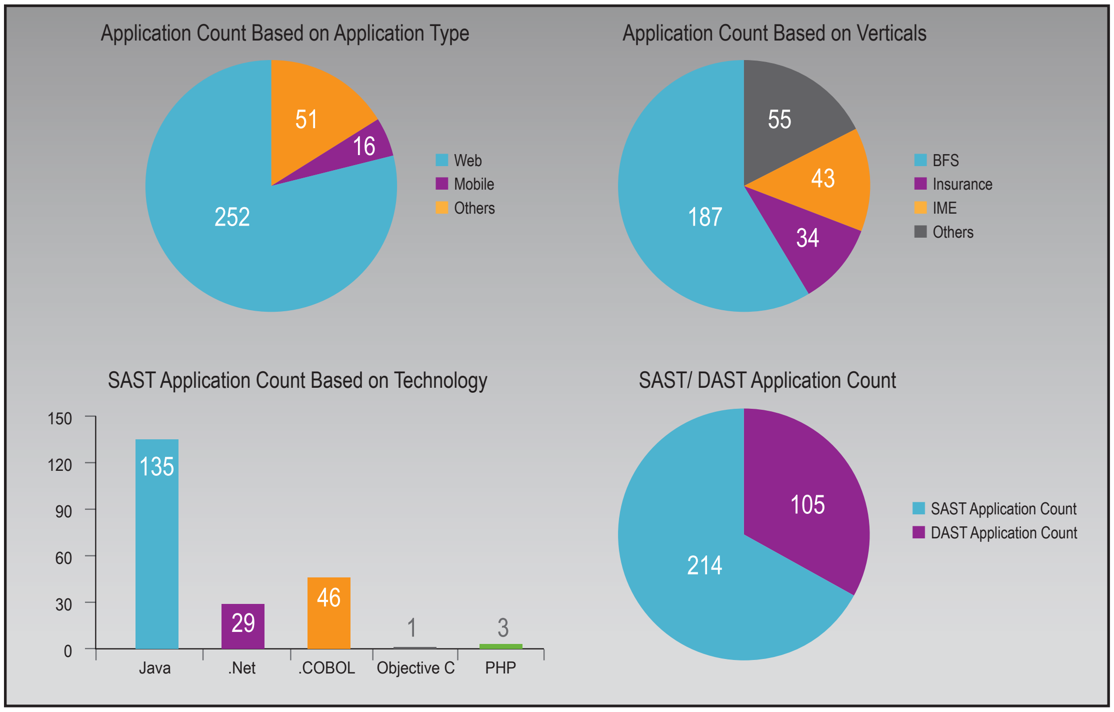


Figure 10

Application code within the BFS domain constitutes the largest portion of applications under security testing — nearly 72%. The application pool comprises 13% of IME applications and 6% insurance, with the remaining being retail, healthcare, travel and hospitality, consumer goods, and manufacturing and logistics applications.

## Glossary

**BFS:** Banking and Financial Services

**CWE:** Common Weakness Enumeration

**DAST:** Dynamic Application Security Testing

**ERSS:** Enterprise Risk and Security Solutions

**HIPAA:** Health Insurance Portability and Accountability Act

**IME:** Information, Media and Entertainment Practice

**OWASP:** Open Web Application Security Project

**PCI:** Payment Card Industry

**SANS:** SysAdmin, Audit, Networking, and Security

**SAST:** Static Application Security Testing

**T&H:** Travel and Hospitality

**WASC:** Web Application Security Consortium

## Footnotes

[1] Regulatory Compliance: Regulations a company must follow to meet specific requirements.

[2] OWASP: Open Web Applications Security Project. A worldwide, not-for-profit charitable organization focused on improving the security of software.

[3] SANS: The SANS Institute was established in 1989 as a cooperative research and education organization.

[4] WASC: The Web Application Security Consortium (WASC) is a 501c3 nonprofit comprising an international group of experts, industry practitioners and organizational representatives who produce open source and widely agreed upon best-practice security standards for the World Wide Web.

[5] CWE: Common Weakness Enumeration provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection and use of software security tools and services that can find these weaknesses in source code and operational systems, and better understand and manage software weaknesses related to architecture and design.

## Acknowledgments

The author would like to recognize the contributions of the following Cognizant associates to this white paper:

Vimalaasree Anandhan, Security Architect
Mahalakshmi Ravi, Security Analyst
Kavitha Karunakaran, Security Specialist
Sandhana Joldrine Xavier, Security Analyst
Saravanan Sankaran, Security Specialist
A, Rajakumari, Security Analyst
K, Ratnadeepika, Security Analyst

The author would also like to thank the following experts and analysts in application security assessment:

A Ajantha
Anjum Afrin
Arul Sumithra
Balasunder Rakesh
Balu Chitra
Celine George
TK Chendhilkumar
Durga Surya Kumar Simma
U Gopinath
S Gowripriya
Jothi Prakash, Grace Catherine
K Subhashini
Deivendran Karthiga
Murugan Vignesh
Muthuramalingam Jose Arokia Mary
Muthuveeran Subash
Narayanan Droupathy Subhash
K Padma Prasoona
Jayaprakash Pavithra
Radhakrishnan Agashnarayani
Ramdass Karthikeyan
Karuppiah Nagamarimuthu
Kuruvilla Mathew
M Balaji Swaminathan
Mantraratnam Sweta
Marreddi Venkatesh

K Rega
S Nisha
Sambasivam Suganiya
Selvaraj Nithya
S Sivapradha
Srinivasan Amith
Sundaram Kalicharan
Sundaramurthy Subhashini
Thomas, Lijo
V Satheesh Kumar
Varadarajan Pradeep
Vedeshwar Raghavendra
GK Yashwanth
Haja Mohideen T Mohaideen Natchiya Sharmeela
Jain Dinesh
Jemmi Angelin

## About the Author

*Dr. Sivakumar Kathiresan, M.E., PhD., is a Principal Architect, Technology, within Cognizant's Enterprise Risk and Security Solutions group. In this role, he leads the Application Security Assessment team, and has managed 120-plus security assessment projects across different verticals over the last three years. Sivakumar has 20 years of experience, including in industry, research and academia, and has delivered more than 100 knowledge-sharing sessions on various fields of enterprise security at different forums. His current areas of interest are advanced log analysis, vulnerability management, advanced persistent threats and management, and security analytics. Sivakumar received his PhD from the Indian Institute of Technology, Roorkee. He continues to research the area of Web security. His certificates include CEH, Sourcefire, Qualysguard, Envision, LanDesk and Big Data Associate. He can be reached at Sivakumar.Kathiresan@cognizant.com.*

## About Cognizant Enterprise Risk and Security Solutions

The Cognizant Enterprise Risk and Security Solutions (ERSS) group specializes in providing end-to-end information-security solutions for various industry verticals, including retail, banking and financial services, logistics, telecom, healthcare, manufacturing and travel and hospitality – having serviced over 400 customers across various geographies. Our team of experts provides information-security solutions and services based on best-of-breed products in each category of enterprise security. Our services encompass:

- 600-plus security consultants specializing in Identify and Access Management (IAM), Governance, Risk and Compliance (GRC), Data Security, Application Security Assessment (Secure SDLC) and Integrated Threat Management.
- 300+ CISA, CISM, CISSP, CEH and vendor-certified associates.
- 250+ Infrastructure Security trained associates.
- Over 7000 combined years of information-security experience.
- A proven track record and experience in 400-plus client engagements for security services.
- Partnership with leading vendors such as IBM, CA, Oracle, SailPoint, Novell, Dell, RSA, HP, Symantec and McAfee.