

WEB APPLICATION VULNERABILITY STATISTICS (2014)



POSITIVE TECHNOLOGIES
2015

Contents

| | |
|---|----|
| Introduction..... | 3 |
| 1. Research methodology..... | 4 |
| 2. Executive summary..... | 5 |
| 3. Participant portrait..... | 6 |
| 4. Vulnerability statistics..... | 8 |
| 4.1. The most common vulnerabilities..... | 8 |
| 4.2. Vulnerabilities in web app development tools..... | 11 |
| 4.3. Vulnerabilities in different web servers..... | 14 |
| 4.4. Statistics by a variety of industries..... | 16 |
| 4.5. Test technique comparative analysis..... | 19 |
| 4.6. Vulnerabilities in test and production apps/ | 20 |
| Conclusion..... | 22 |
| References..... | 23 |

Introduction

Many large corporations now rely on entwined web applications to run their business including official websites, e-commerce and e-banking platforms, online shops, etc. Many these web applications are no longer hosted by dedicated client software but in cloud services with advanced web technologies. While these web services are now crucial for business, they are also very vulnerable. Compromised applications can trigger reputation damage, loss of important clients, financial loss and increase operational expenses when time and money must be spent to recover affected resources. That is why application security is no less essential than the development of the application's functions.

However, developers and administrators are rarely skilled in information security and make mistakes that expose applications to attacks. Overwhelmingly, web resource owners do not follow secure software development lifecycle procedures, and vulnerabilities remain unnoticed even when an application is live, playing into attackers' hands.

This research study reviews and analyzes the most widespread web application vulnerabilities found during a series of tests carried out by Positive Technologies in 2014.

¹ <http://fortune.com/global500/>

² http://expert.ru/ratings/rejting-krupnejshih-kompanij-rossii-2014-po-ob_emu-realizatsii-produktsii/

1. Research Methodology

This publication provides an overview of the web application vulnerability statistics gathered during penetration testing, security audits and other work performed by Positive Technologies in 2014. In total, the experts at Positive Technologies examined around 300 unique web applications using various techniques from instrument to source-code analysis. To obtain an objective sample, we chose 40 applications, from the 300 examined, and conducted an in-depth analysis on each one of these.

These 40 sites contained 1,194 vulnerabilities of various severity levels. The security level of each application was estimated using tests conducted by automated tools and also manually using a combination of black-, gray- and white-box methods. Black-box analysis estimates an application's security level from the perspective of an external attacker, with no "inside" knowledge of the application. Gray-box testing is similar to black-box, except an attacker is considered a user with some system privileges, like an employee. White-box scanning requires the tester to have all relevant information about the application, including its source code, and usually provides a truer security picture. The vulnerability statistics included in this report are only for external web applications that are accessible over the Internet.

The vulnerabilities found were classified according to the corresponding threats in the Web Application Security Consortium Threat Classification (WASC TC v. 2 [1]), with the exception of Improper Input Handling and Improper Output Handling, as these threats are implemented by exploiting a number of other vulnerabilities.

Our results include only code and configuration error vulnerabilities. Other widespread information security weaknesses such as using old versions of software or results obtained in the course of instrument scanning and penetration testing are not considered in this report.

The severity level of these vulnerabilities was calculated according to the Common Vulnerability Scoring System (CVSS v. 2 [2]). Based on the CVSS score obtained, our experts assigned vulnerabilities with severity levels high, medium or low.

2. Executive Summary

This section summarizes the most important findings of the 2014 report.

1. All web applications analyzed have vulnerabilities. **High-severity vulnerabilities were detected in 68% of systems**, an increase from 2013 (62%). In 2013, web applications had 15.6 vulnerabilities on average; in 2014, this number almost doubled to 29.9. Specifically in terms of high-severity vulnerabilities discovered by black- and gray-box tests, the average number of vulnerabilities increased from 2.8 to 7.5.
2. **In 2014, the most common vulnerability was Fingerprinting**, present in 73% of the applications. In 2013, this security flaw was detected in 49% of web resources and was the fourth most common. Cross-Site Scripting, most common in 2013, was third in 2014, found in 70% of systems studied. If either of these flaws are exploited, an attacker can gain access to user's personal account.
3. **The Top 10 errors found also include SQL Injection**, fourth most common, with **48%** of the analyzed systems affected. In 2013, this code flaw was the sixth most widespread — detected in 43% of websites. These exploits allow unauthorized access to sensitive information stored in application databases and could allow an attacker to gain full control over a target server.
4. The most vulnerable websites are **banks' web applications. 89% of their systems were exposed to critical vulnerabilities**. There is some bias in the sample pool used as the majority of the resources tested were not online services or other systems that handle cash transactions, so they may not feature the highest levels of data security. In 2013, the comparable index was 67%, and banking applications were more profoundly protected in comparison to IT and mass media apps.
5. **Of applications written in PHP, 81% contain critical vulnerabilities**. An average PHP application contained 11 critical vulnerabilities in 2014, similar to the results of 2013, and 31 medium-severity vulnerabilities, 2.5 times higher than in 2013.
6. **In 2014, Nginx proved to be the most vulnerable web server**: 86% of web resources run by Nginx servers contained high-severity vulnerabilities. In 2013, high-severity vulnerabilities occurred on just 57% of sites. In addition, seven out of ten Apache applications were also exposed to critical vulnerabilities.
7. **White-box testing revealed 3.5 times more medium-severity vulnerabilities** than black- and gray-box testing. Each site tested with black- and gray-box testing methods uncovered 4 XSS vulnerabilities compared to 29 when employing a white-box testing techniques, demonstrating the additional vulnerabilities to systems in terms of internal attack.
8. **71% of the production web resources** and 50% of the test sites surveyed contained **critical vulnerabilities**. The average number of these high level severity vulnerabilities detected in the test systems, 12.8, is almost twice as high as production ones, 7; however, the latter contain larger number of medium-severity vulnerabilities (20.6 vs 14.3). This demonstrates the importance of secure software development lifecycle processes.

3. Participant Portrait

This study includes companies from a wide range of industries including manufacturing, banking, e-commerce, telecoms, IT, and one government-owned institution. As the government sector was represented by a single application, its detailed statistics are omitted from this report.

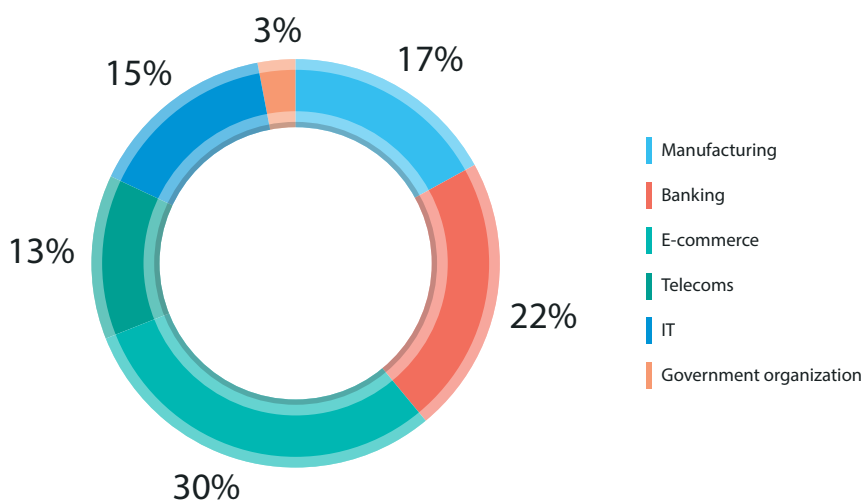


Figure 1. Systems by industry

The web resources analyzed were written in various programming languages with the use of different technologies. Among the systems studied, PHP- and ASP.NET-based web applications were the most common. Other languages and technologies, such as JSP, JSF, Java SE/EE, and Perl, were small in number in 2014 and are grouped together in the category "Other".

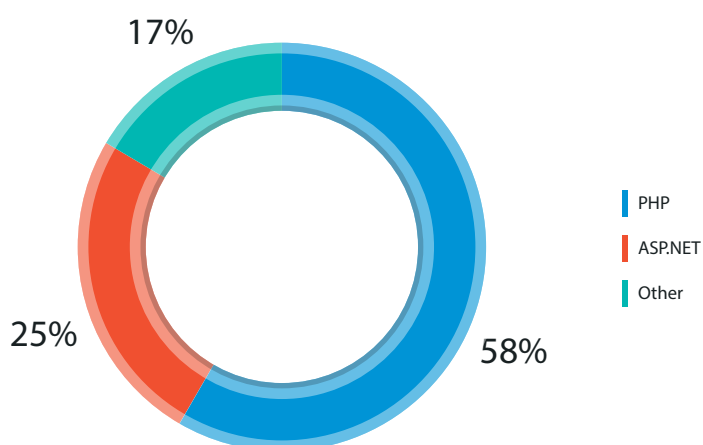


Figure 2. Systems by development technologies

Based on the web server used, all web applications studied were subdivided into four groups: run by Apache, Microsoft Internet Information Services (IIS), Nginx, and other web servers (Apache Tomcat and Apache Coyote). The most widely used web server was Nginx: 37% of web applications were based on it.

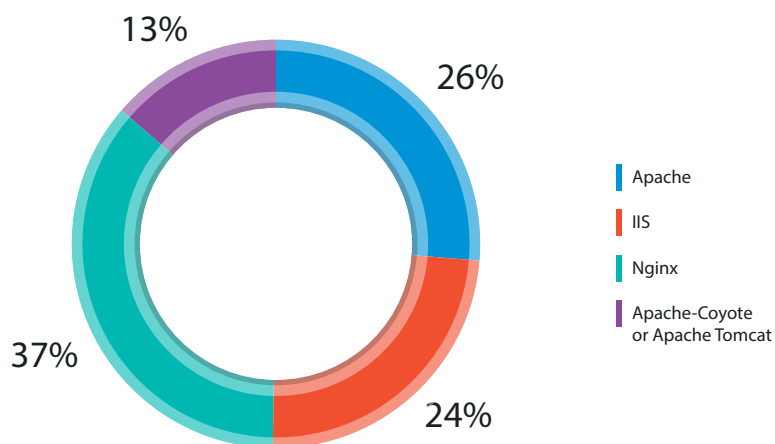


Figure 3. Systems by web servers used

Web applications studied include both production systems available over the Internet and test beds under development or acceptance for operation. Production systems are the largest segment of the sample pool and make up 85%.

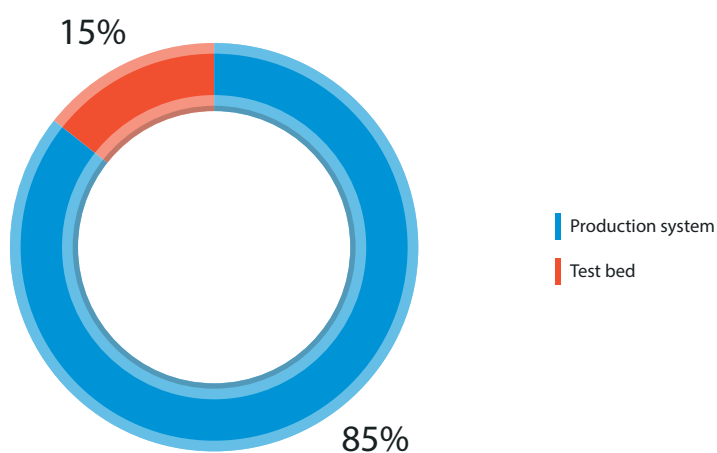


Figure 4. Systems by environment

4. Vulnerability Statistics

As in 2012 and 2013, commercial content management systems (CMS) were rarely used. As a result, there were no statistical investigations on the security level of web sites that depend on a CMS.

This section includes analysis regarding the frequency of various vulnerabilities and their associated severity levels, based on the threats described in WASC TC v.2.

4.1. The most common vulnerabilities

According to our research, in 2014, 68% of web resources contained critical vulnerabilities, higher than the 62% in 2013.

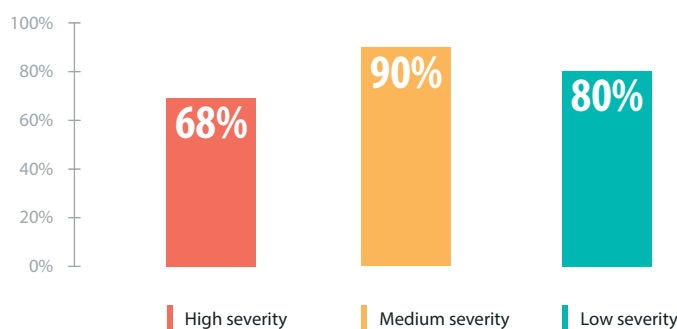


Figure 5. Websites by vulnerability severity

All the web applications tested in 2014 are vulnerable.

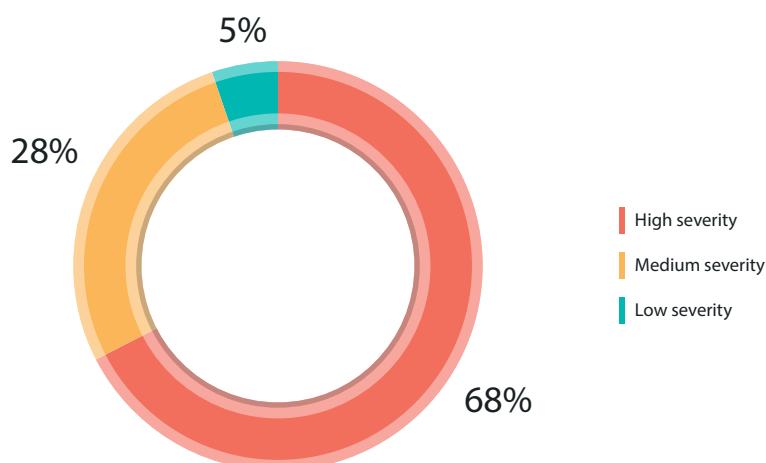


Figure 6. Websites by maximum severity

The percentage of systems with high-severity security flaws increased from 2013 to 2014. The percentage of systems with low-severity vulnerabilities increased to 80% while those with medium-severity — dropped to a similar percentage as 2012 (90%).

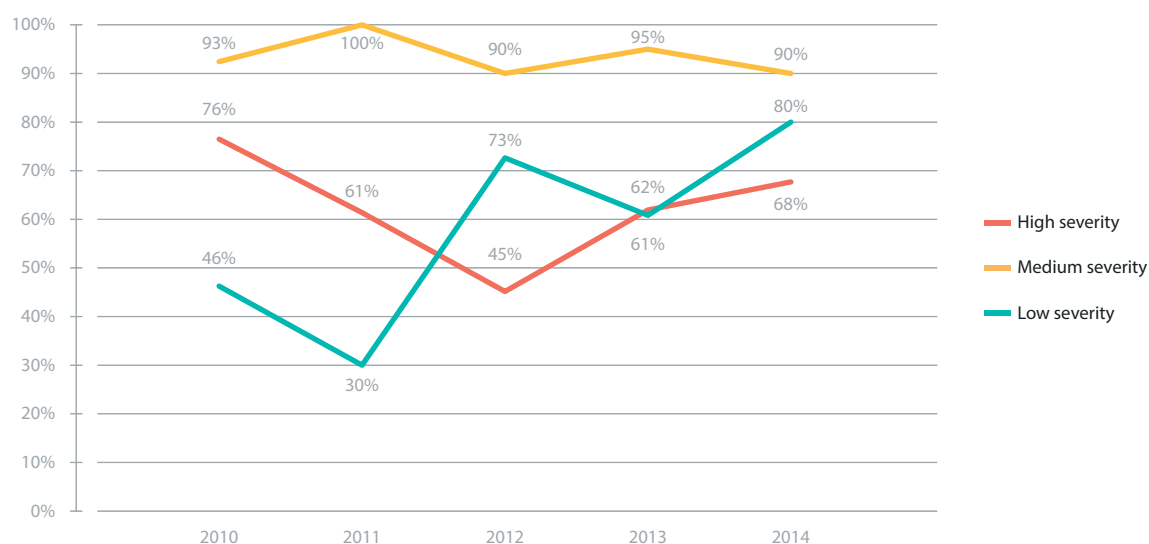


Figure 7. Websites by vulnerability severity

In 2014, a low-severity vulnerability called Fingerprinting spread to almost three out of four resources (73%). The most common vulnerability of 2013, Cross-Site Scripting (XSS), was second most common in 2014, affecting 70% of web applications analyzed. This security weakness allows attackers to inject arbitrary HTML tags, including JavaScript, into a browser and obtain a session ID to access the application and its privileges. More than a half of websites contained vulnerabilities related to predictable user and session IDs (Credential/Session Prediction).

The ten most common vulnerabilities also include SQL Injection, a critical vulnerability detected in 48% of the web resources studied. If successfully exploited, this vulnerability provides for unauthorized access to the DBMS and confidential information stored in the database. If a DBMS account has necessary privileges on the OS, attackers can even obtain full control over the server.

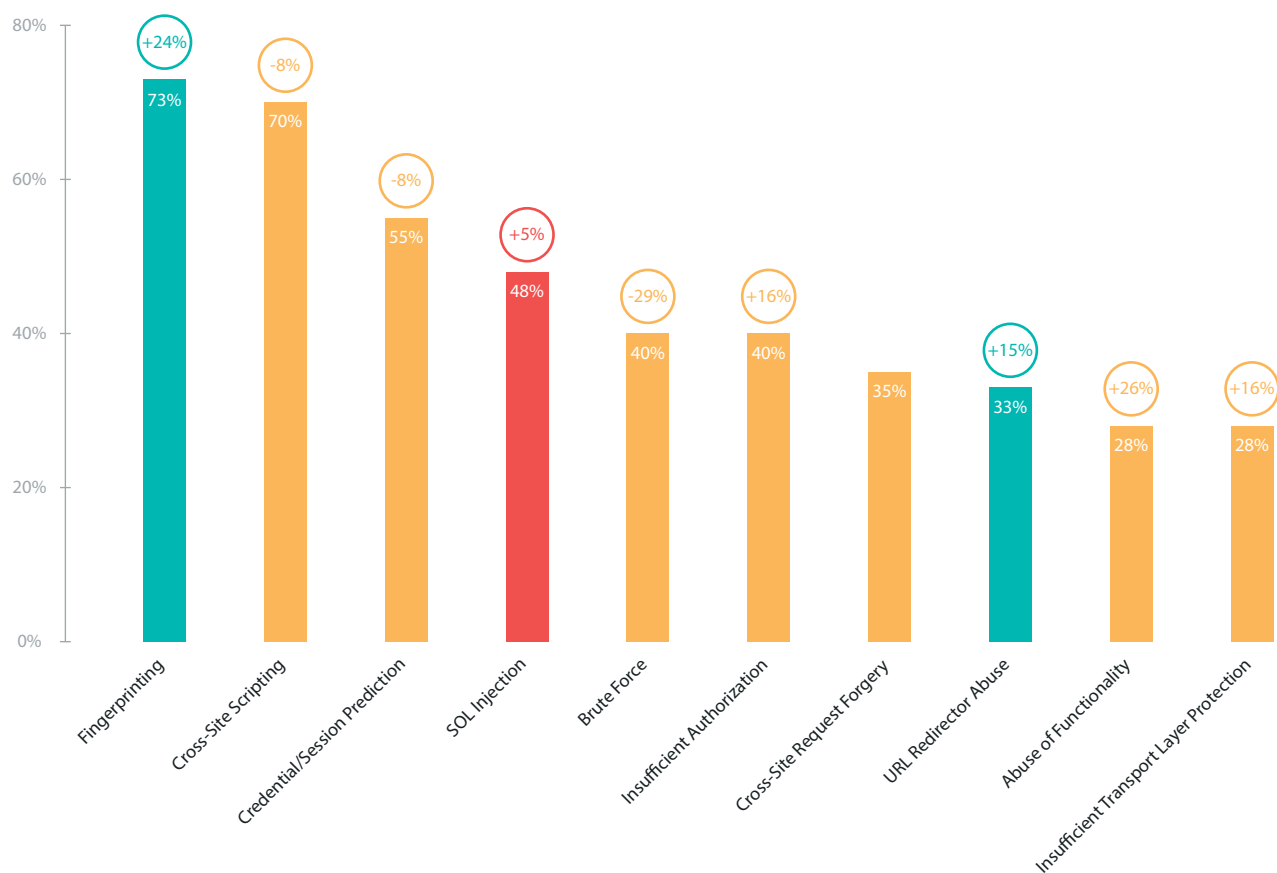


Figure 8. Most common vulnerabilities (%)

A majority (54%) of security weaknesses found allow hackers to attack web application clients. As in 2013, Cross-Site Scripting and Cross-Site Request Forgery are the most common examples.

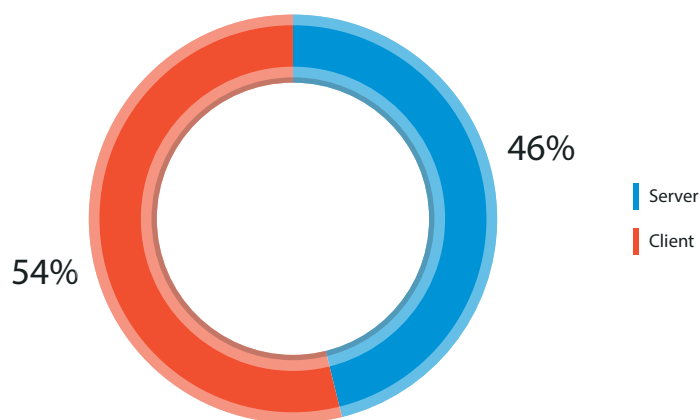


Figure 9. Application vulnerabilities by attack targets

A large majority (89%) of vulnerabilities were found in software code; while only 11% resulted from web application misconfiguration.

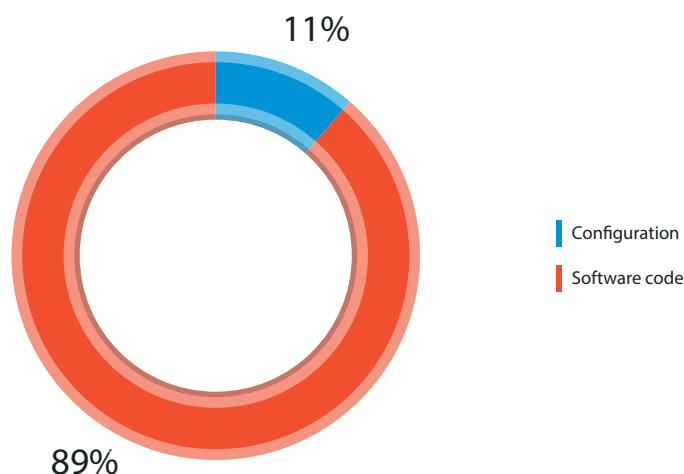


Figure 10. Application vulnerabilities by source

4.2. Vulnerabilities in web app development tools

This section includes statistics on vulnerabilities that are common in a variety of web app development tools. For sites written in PHP, an overwhelming 81% showed critical flaws. ASP.NET vulnerabilities decreased from 55% in 2013 and made up only 44% in 2014. High-severity vulnerabilities of applications developed with other tools (JSP, JSF, Java SE/EE, and Perl) together amount to 83%.



Figure 11. Systems with vulnerabilities of various severity levels (by development tools)

All resources included in the study contained vulnerabilities. Applications based on technologies other than ASP.NET were found to have vulnerabilities of at least a medium severity level.

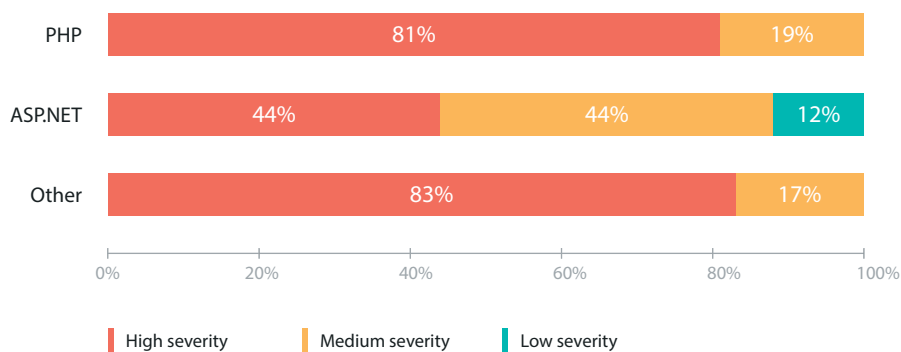


Figure 12. Systems by maximum vulnerability severity

On average a PHP application contained 11 critical vulnerabilities, while an ASP.NET app had 8.4. These statistics are heavily skewed by one outlier, an ASP.NET system that had 60 high-severity flaws. If this outlier case is excluded, the average number of critical vulnerabilities found drops to only 2 vulnerabilities per application. In 2013, this value was almost the same (1.6).

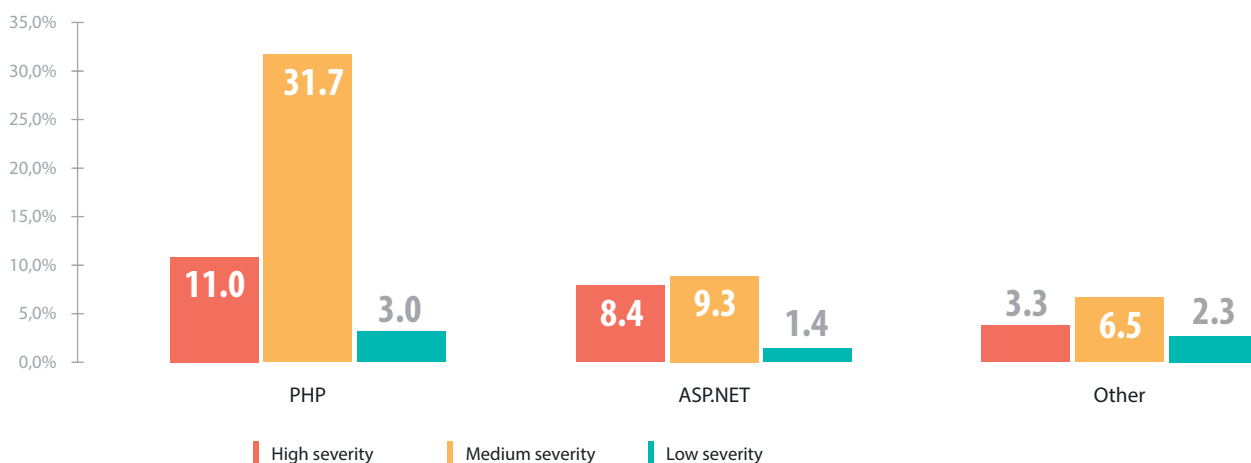


Figure 13. Average number of vulnerabilities per system by development tools

Table 1 includes statistics on the frequency of common vulnerabilities among resources developed by different tools.

| PHP | % of websites | ASP.NET | % of websites | Other | % of websites |
|-------------------------------|---------------|---|---------------|-------------------------------|---------------|
| Cross-Site Scripting | 95 | Fingerprinting | 78 | Fingerprinting | 67 |
| Fingerprinting | 76 | Cross-Site Scripting | 44 | Credential/Session Prediction | 67 |
| SQL Injection | 67 | Insufficient Authorization | 44 | Cross-Site Scripting | 50 |
| Credential/Session Prediction | 62 | Brute Force | 44 | Brute Force | 50 |
| Abuse of Functionality | 48 | SQL Injection | 33 | Insufficient Authorization | 33 |
| Insufficient Authorization | 43 | Credential/Session Prediction | 33 | SQL Injection | 33 |
| Cross-Site Request Forgery | 43 | XML External Entities | 33 | Cross-Site Request Forgery | 33 |
| URL Redirector Abuse | 43 | Abuse of Functionality | 22 | URL Redirector Abuse | 33 |
| Brute Force | 38 | Insufficient Transport Layer Protection | 22 | Information Leakage | 33 |
| Information Leakage | 33 | Path Traversal | 22 | Denial of Service | 33 |

Table 1. Most common vulnerabilities

Cross-Site Scripting was the most common vulnerability for the PHP programming language — 95% of resources suffered from it. Fingerprinting, a low-severity flaw, was the most wide spread vulnerability among the other development tools. SQL injection, a critical vulnerability, appeared in 67% of sites written in PHP. For the other tools, this index was lower (33%), though the percentage of applications exposed to this security flaw increased significantly.

The figures below provide the percentage of web resources with vulnerabilities widespread in 2014.

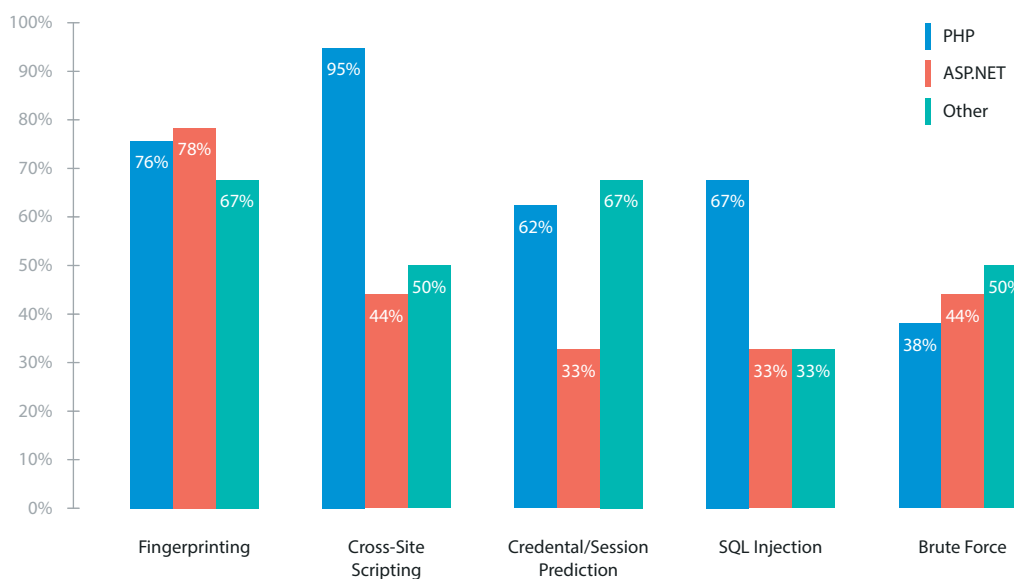


Figure 14. Applications with the most widespread vulnerabilities by development tools (part 1)

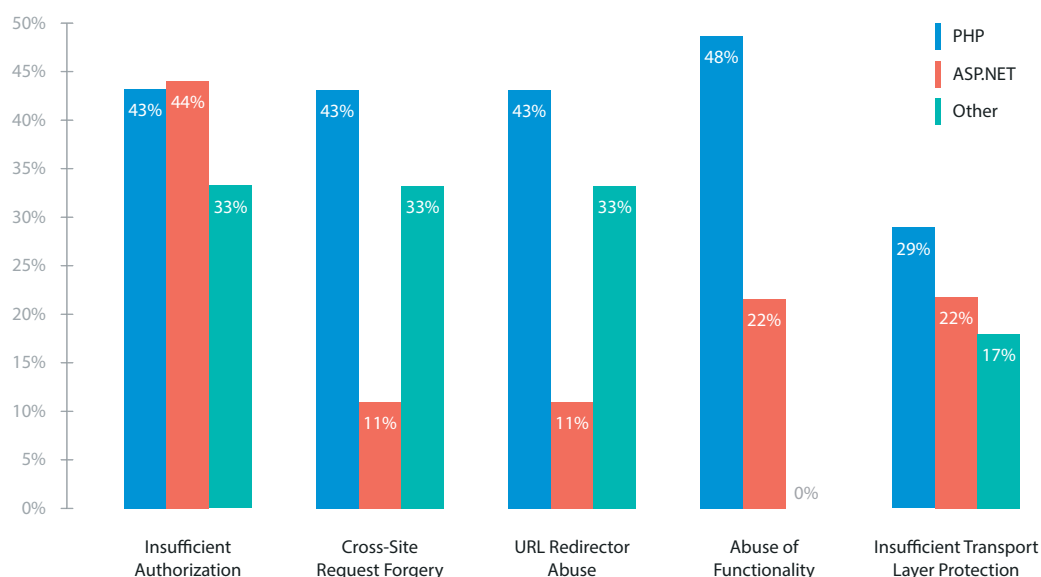


Figure 15. Applications with the most widespread vulnerabilities by development tools (part 2)

PHP-based resources are more vulnerable to Cross-Site Scripting than ASP.NET sites because the latter have built-in protection mechanisms such as request validation.

4.3. Vulnerabilities in different web servers

The security analysis reveals that 86% of Nginx-run web applications had high-severity vulnerabilities. The web applications run on Microsoft ISS-based resources had similar vulnerabilities in 44% of cases, a decrease from 2013 where these incidents occurred in 71% of cases. By contrast, the vulnerabilities in Apache sites increased from 2013 to 2014 by 10% and amounted to 70% in 2014.

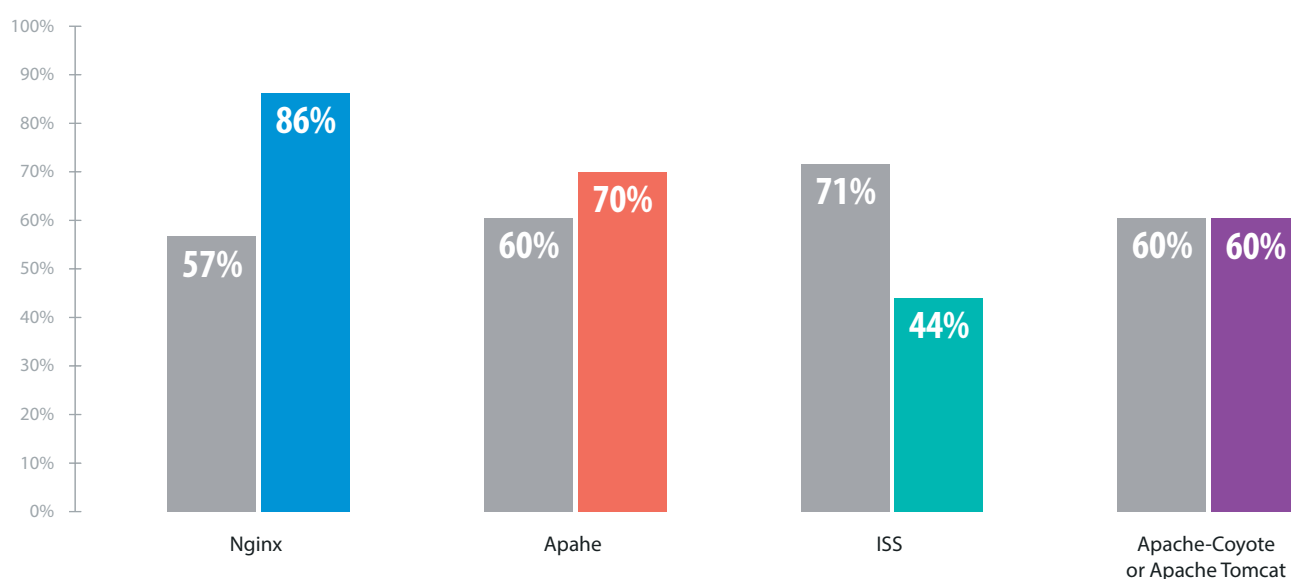


Figure 16. Web applications with high-severity vulnerabilities (by web servers)

Some vulnerabilities classified according to WASC TC v. 2 are the result of improper administration of web applications. Vulnerability statistics for web servers most frequently used in the systems studied are provided below.

| Apache | % of websites | IIS | % of websites | Nginx | % of websites | Apache-Coyote or Apache Tomcat | % of websites |
|---|---------------|---|---------------|---|---------------|---|---------------|
| Fingerprinting | 80 | Fingerprinting | 78 | Fingerprinting | 79 | Fingerprinting | 60 |
| Brute Force | 40 | Brute Force | 44 | Brute Force | 36 | Brute Force | 60 |
| Information Leakage | 30 | Insufficient Transport Layer Protection | 22 | Insufficient Transport Layer Protection | 29 | Information Leakage | 60 |
| Insufficient Transport Layer Protection | 30 | Server Misconfiguration | 22 | Information Leakage | 21 | Insufficient Transport Layer Protection | 40 |
| Server Misconfiguration | 20 | Information Leakage | 11 | Predictable Resource Location | 14 | Insufficient Session Expiration | 20 |

Table 2. Rating of administration vulnerabilities for different web servers

On all the web servers tested, the most common administration flaw was Fingerprinting, present in eight out of ten Apache-based resources. The cause of this vulnerability is that standard configuration of the examined servers allows for disclosure of information about a server version through error messages (for example, when calling to a nonexistent resource).

Figure 17 below compares the percentage of vulnerable resources administrated by different web servers for every administration flaw.

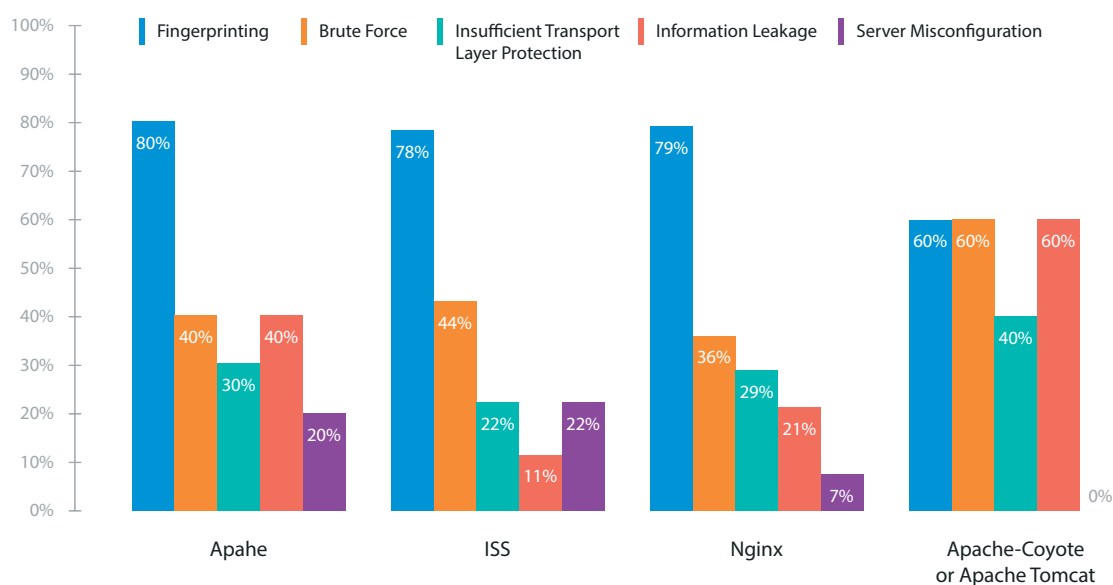


Figure 17. Vulnerable sites on a variety of web servers

4.4. Statistics by a variety of industries

This section considers web applications used in various industries including manufacturing, banks, telecommunications, e-commerce, and IT. In 2014, the government sector was represented by a single application, therefore, its statistics are excluded. But it is worth noting that this web application was found to have critical vulnerabilities in the government sector example.

Bank-owned applications are the most vulnerable in 2014 as 89% of sites are exposed to critical vulnerabilities. Traditionally, banks devote much attention to security of data they process; nonetheless, their applications were found to include high-severity vulnerabilities. This result might be exacerbated by the testing pool used, as the largest part of the resources tested were not online services or other systems that handle money transactions. Banks may neglect the security of their data in the non cash transaction applications.

Many web applications exposed to critical vulnerabilities were also found in the telecoms industry. In 2012, the majority of systems with high-severity flaws were detected in this industry, but in 2013 only two applications were studied so due to this lack of data this research omits comparison of 2014 and 2013 results.

The least vulnerable were e-commerce applications, though they contained 42% of critical weaknesses.

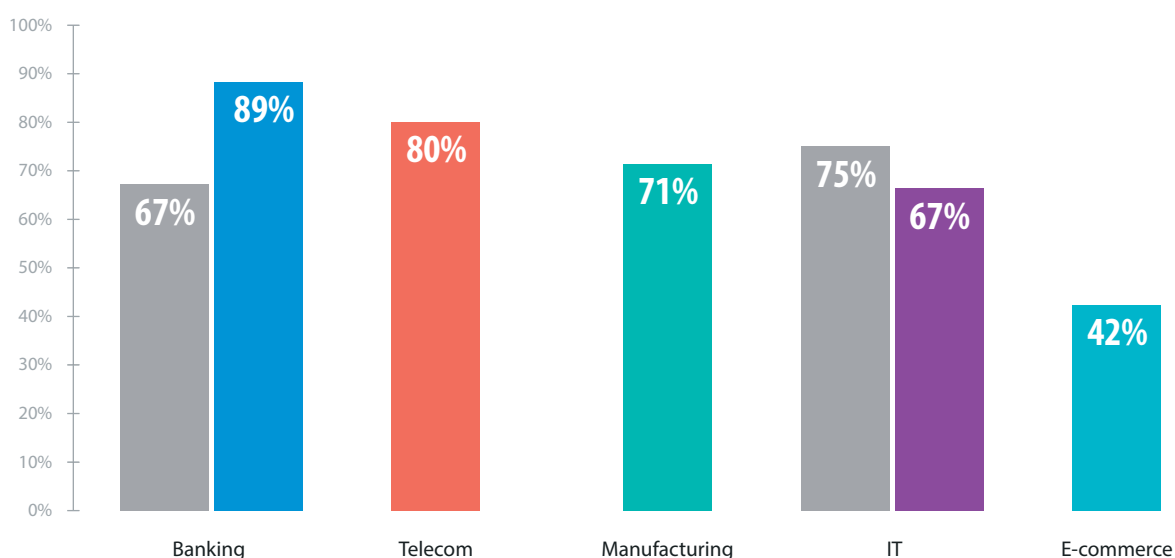


Figure 18. Sites with high-severity vulnerabilities by industries

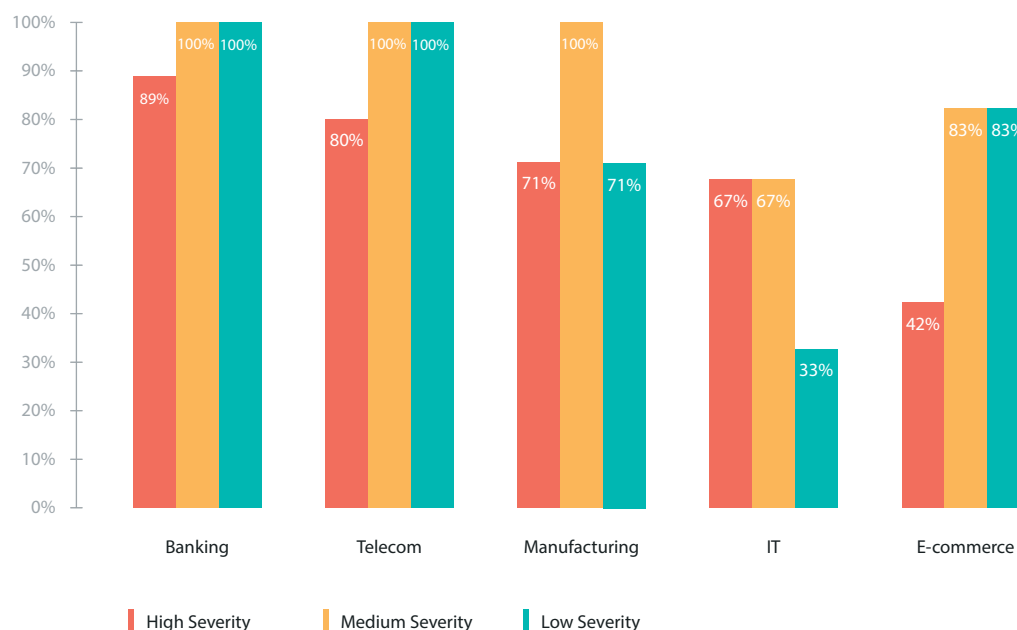


Figure 19. Sites exposed to vulnerabilities of a specific severity level for each industry considered

Based on the average number of vulnerabilities per system, the least protected sites were in the manufacturing industry with 18 critical flaws per application. It is worth noting that the aforementioned application with 60 vulnerabilities was from the manufacturing sector. If that outlier is removed, the average number drops to 13.1, which mimics the rate in banking.

The average system in the banking sector had 13 critical and 56 medium-severity flaws on average, which was due to erroneous PHP implemented in the majority of systems tested in 2014 (see section 4.2). In addition, white-box testing detected 388 medium-severity vulnerabilities in one application affecting the fair value for the industry.

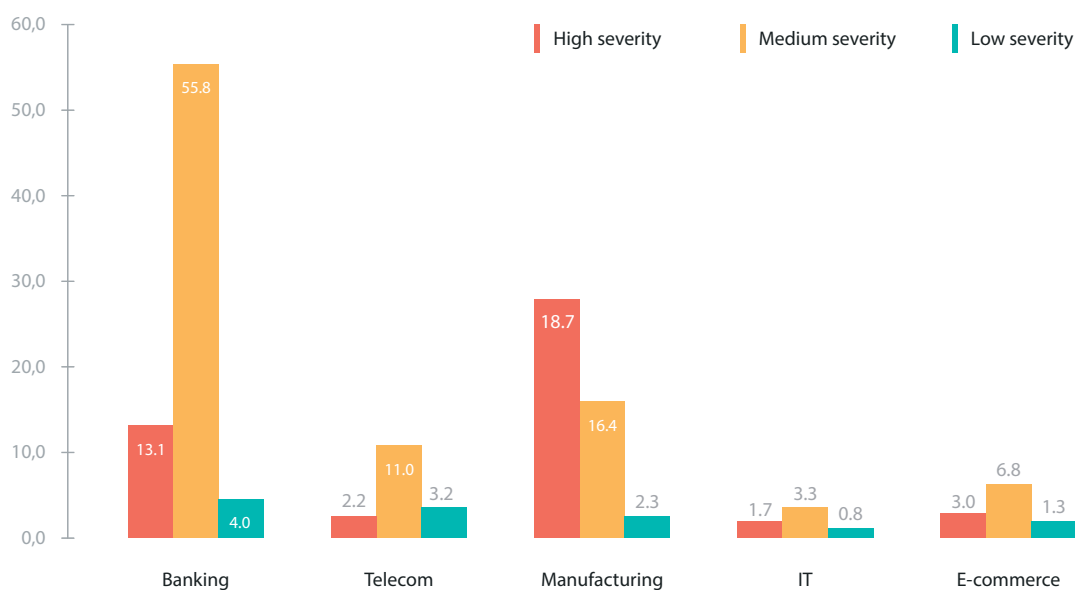


Figure 20. Average number of vulnerabilities per system

In 2014, the most common critical vulnerabilities were SQL Injection, XXE Injection and Directory Traversal. All three were discovered in the web resources of banks. Similar to the previous year, the SQL Injection flaw was present in web applications from all industries.

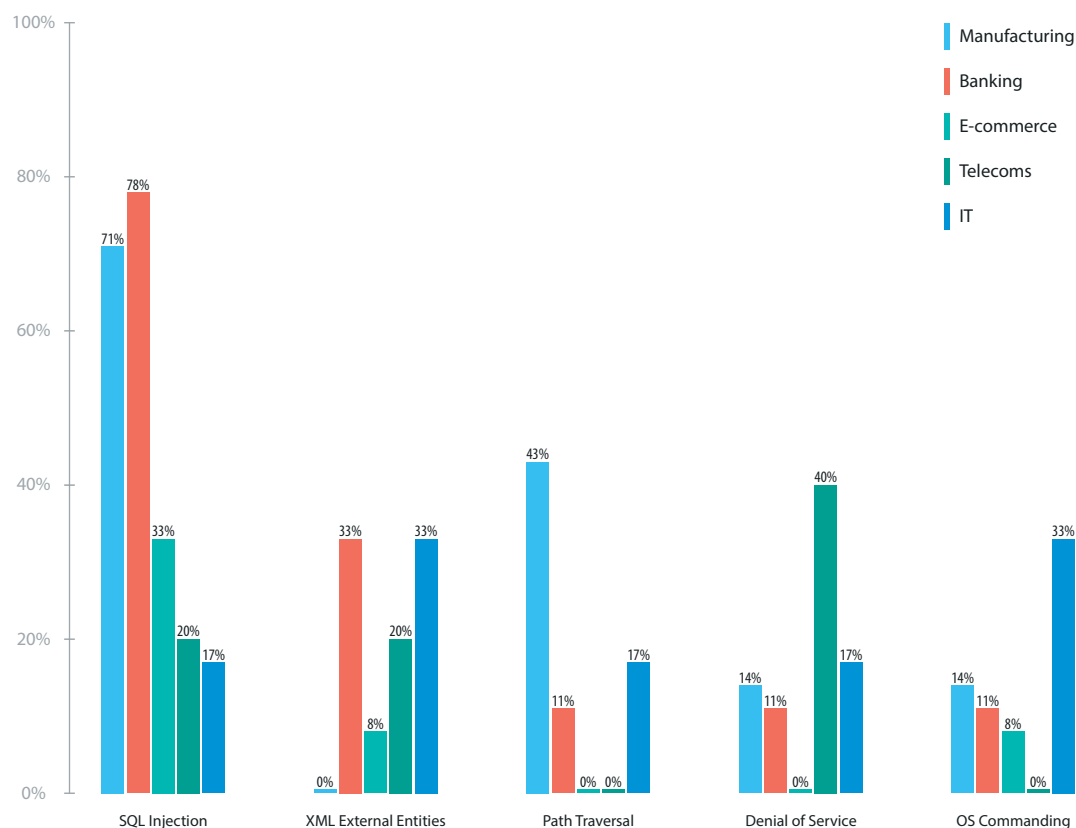


Figure 21. Vulnerable sites in a variety of industries

The research results show that the majority of systems in all the industries, except e-commerce, were exposed to high-severity vulnerabilities. However, even e-commerce apps contained dangerous flaws in 42% of cases (42% medium-severity and 17% low-severity security bugs).

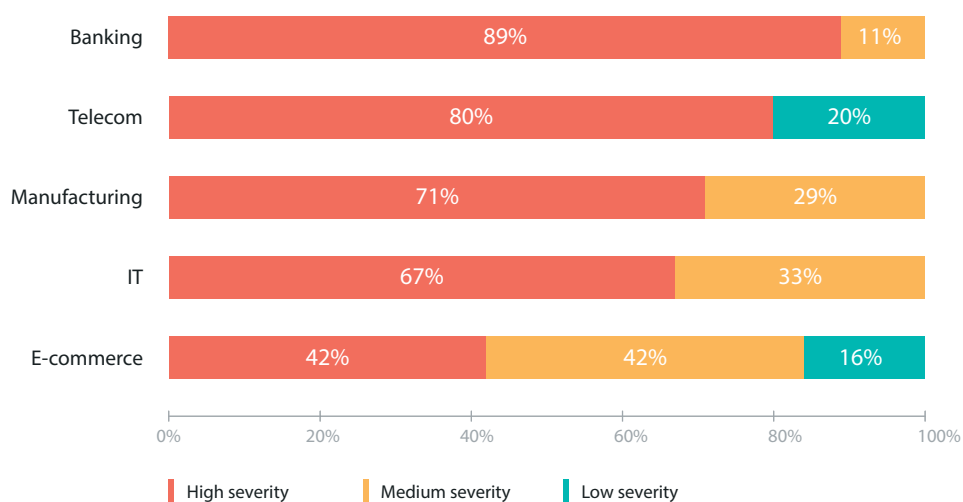


Figure 22. Systems by maximum vulnerability severity

4.5. Test technique comparative analysis

Positive Technologies experts compared the results of white-box testing and the results that came from black- and gray-box testing. The percentage of sites containing high- and medium-severity flaws was similar across all three testing methodologies. A lack of source code does not stop hackers from attacking web applications as very often extra knowledge of the system is not required to detect and exploit high- and medium-severity vulnerabilities.

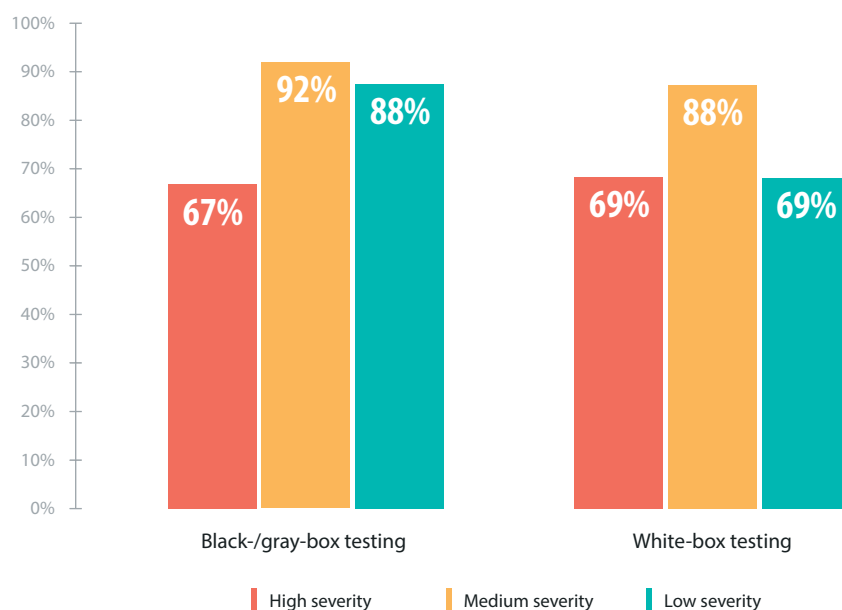


Figure 23. Systems with different severity vulnerabilities by test technique

Comparing the average number of vulnerabilities found per system shows that white-box testing detects more vulnerabilities regardless of their severity; in particular, white-box testing discovered 3.5 times more medium-severity flaws on average compared to black- and gray-box testing methods.

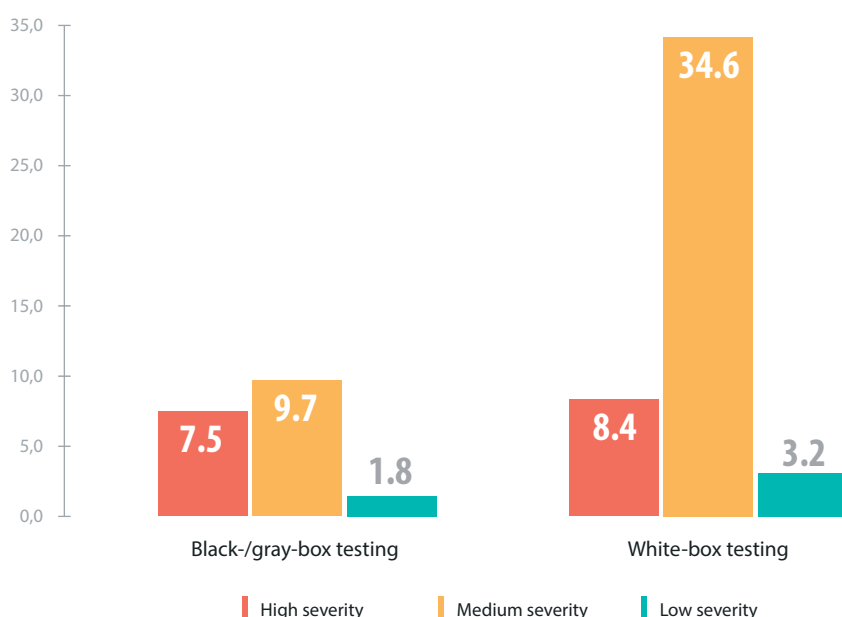


Figure 24. Number of vulnerabilities per system by their testing method

For example, each site tested with black- and gray-box testing methods uncovered roughly 3.9 XSS vulnerabilities compared to 29.1 when employing a white-box testing method.

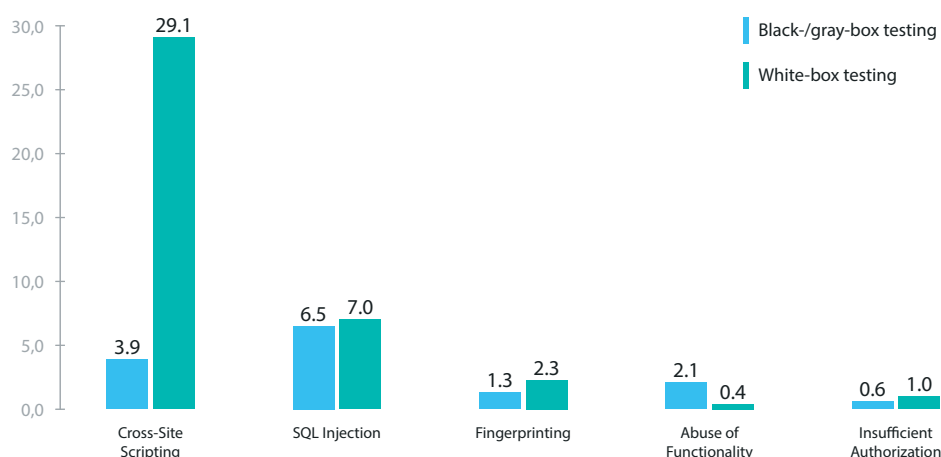


Figure 25. Number of certain vulnerabilities per system by their testing method

These results argue for source code analysis conducted alongside black- and gray-box testing techniques for better detectability. Gray- and black-box techniques are limited in what they will find, since auditors may fail to detect existing vulnerabilities fearing service denial. Hackers, by contrast, have wider opportunities to accelerate DoS attacks.

4.6. Vulnerabilities in test and production apps

This section provides the results of security analysis for test and production systems. Figure 26 shows the percentage of different severity vulnerabilities in systems and test beds that have already been deployed.

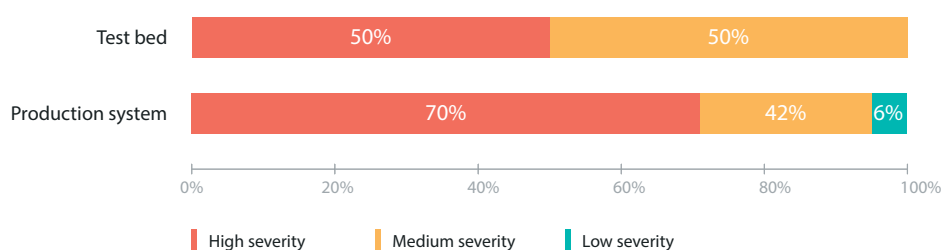


Figure 26. Vulnerabilities detected for test and production systems

71% of the production web resources and 50% of the test sites surveyed contained critical vulnerabilities. This is a strong indicator that production apps are not necessarily more secure than test beds.

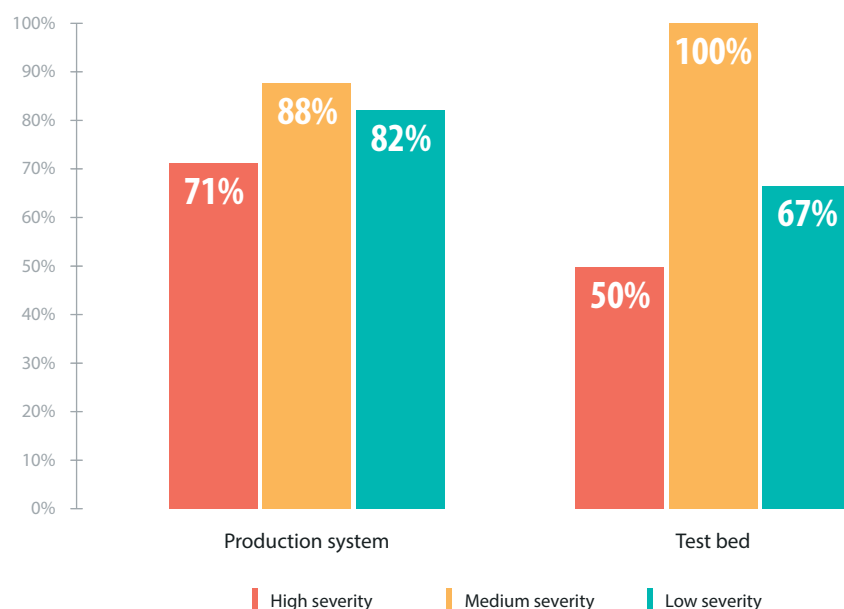


Figure 27. Vulnerable sites in test and production systems

The average number of high-severity vulnerabilities detected in test systems is 12.8, almost twice as many as in production applications; however, the latter contain a larger number of other vulnerabilities involving those of medium severity and remain exposed to hacks. These vulnerabilities in systems already in production indicates that secure software development lifecycle processes (secure SDLC) is not being followed. Secure app development will reduce the number of weaknesses available to attackers.

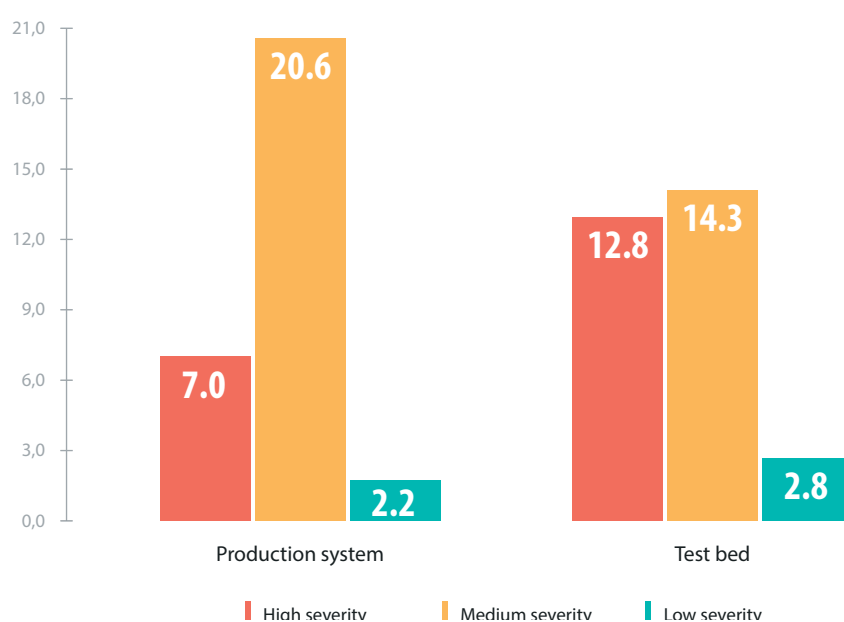


Figure 28. Average number of vulnerabilities per application in test and production systems

Conclusion

This research argues that web applications remain vulnerable in 2014, and in fact may be more vulnerable than in 2013. The percentage of sites with critical vulnerabilities increased by six percentage points to 68%.

In 2013, web applications had, on average, 15.6 vulnerabilities while in 2014 this average has almost doubled to 29.9. This growth is partially due to the more reliable white-box testing implemented in a larger number of systems in 2014. But even in systems tested by black- and gray-box techniques, the average number of vulnerabilities per application still increased. For example, this index for high-severity vulnerabilities increased from 2.8 to 7.5 average vulnerabilities per application.

The percentage of resources with medium-severity security flaws dropped to 90%. Production systems were more exposed to such weaknesses than test beds, which indicates security is neglected throughout the lifecycle of products.

Only one site tested had a web application firewall (WAF), so despite many vulnerabilities that could be detected and prevented by WAF, that product is not normally being used to protect web applications.

This research demonstrates that web application security problems still remain overlooked. To reduce risks related to application vulnerabilities, vendors should implement secure development procedures, provide comprehensive testing at the acceptance stage, and use preventive protection measures.

References

1. WASC Threat Classification v. 2.0: <http://projects.webappsec.org/Threat-Classification>.
2. Common Vulnerability Scoring System: <http://www.first.org/cvss>.
3. OWASP Top Ten Project: https://www.owasp.org/index.php/OWASP_Top_Ten_Project.

About Positive Technologies

Positive Technologies is a leading provider of vulnerability assessment, compliance management and threat analysis solutions to more than 1,000 global enterprise clients. Our solutions work seamlessly across your entire business: securing applications in development; assessing your network and application vulnerabilities; assuring compliance with regulatory requirements; and blocking real-time attacks. Our commitment to clients and research has earned Positive Technologies a reputation as one of the foremost authorities on SCADA, Banking, Telecom, Web Application and ERP security, and distinction as the #1 fastest growing Security and Vulnerability Management firm in 2012, as shown in an IDC report*. To learn more about Positive Technologies please visit www.ptsecurity.com.

*Source: IDC Worldwide Security and Vulnerability Management 2013-2017 Forecast and 2012 Vendor Shares, doc #242465, August 2013. Based on year-over-year revenue growth in 2012 for vendors with revenues of \$20M+.

© 2015 Positive Technologies. Positive Technologies and the Positive Technologies logo are trademarks or registered trademarks of Positive Technologies. All other trademarks mentioned herein are the property of their respective owners.

