SECURITY TRENDS & VULNERABILITIES REVIEW

WEB APPLICATIONS

2016

POSITIVE TECHNOLOGIES

Contents

Introduction

Web technologies can be found in many organizations. We extensively use e-commerce and e-banking platforms, online shops, business applications, and other web resources. Very few modern large corporations or small private companies function without an official website or a page on public web resources. Most companies use web technologies that make business processes more effective rather than corporate applications which need clients software and regular update.

To maximize the efficiency of web technology, you need to make resources available for the target audience, usually from the Internet. Compromised applications can trigger reputation damage, loss of important clients, and financial loss, as competitors or intruders may try to steal money and sensitive data or to violate access to resources.

Developers pay more attention to the functionality rather than to the security of web applications, while administrators are rarely skilled in information security and make mistakes that expose applications to attacks. Frequently, web resource owners do not follow secure software development lifecycle procedures, and vulnerabilities remain unnoticed even when an application is out on the market.

Positive Technologies experts examine around 300 web applications every year using various techniques from instrument to source code analysis. This report provides statistics that were gathered during web application security assessments performed by Positive Technologies in 2015. It also compares 2015 results to those in 2013 and 2014. Thus it is possible to track the dynamics of web application development in the context of delivering information security.

1. Research Methodology

We chose 30 applications from those examined in 2015 and conducted an in-depth analysis on each one of them. We excluded results of the web application security analysis obtained in the course of automated scanning or penetration testing. The statistics include results obtained during source code analyzing. The study contains vulnerabilities tested in the testbeds.

The vulnerability assessment was conducted via black-, gray- and white-box testing manually (with the aid of automated tools) or using automated source code analyzer. The black-box technique is defined as website security testing from the perspective of an external attacker, with no "inside" knowledge of the system. The gray-box testing is similar to the black-box testing, except an attacker is defined as a user who has some privileges in the system. The white-box scanning presupposes the use of all relevant information about the application, including its source code.

Vulnerabilities were categorized according to Web Application Security Consortium Threat Classification (WASC TC v. 2), with the exception of Improper Input Handling and Improper Output Handling, as these threats are implemented by exploiting a number of other vulnerabilities.

Our statistics only include code and configuration vulnerabilities. Other widespread information security weaknesses such as improper software updating were not considered.

The severity of vulnerabilities was estimated in accordance with Common Vulnerability Scoring System (CVSS v. 2). Based on the CVSS score obtained, our experts assigned vulnerabilities with the severity levels: high, medium, or low.

2. Executive Summary

This section summarizes the most important findings of this report.

All web applications analyzed have vulnerabilities.

All applications contained at least medium-severity vulnerabilities. 70% of the systems studied had a critical vulnerability. The percentage of systems with this type of vulnerability has grown consistently over the last three years. These figures demonstrate that developers pay little attention to the code quality while administrators are not focused on the security of application configuration.

Application users are not protected.

Most of the applications examined allow users to be attacked. For instance, 80% of the investigated resources were vulnerable to Cross-Site Scripting (XSS) attacks, while 30% were exposed to Open Redirect (or URL Redirector Abuse).

Information disclosure is still in use.

The number of applications, where software version information was disclosed, decreased significantly in 2015. However, any intruder can obtain other sensitive data (application source code, installation path, personal data, etc.). About half of applications are vulnerable to Information Leakage.

Java applications are vulnerable as well.

The majority of applications examined in 2015 were written in Java and only 30% in PHP. Previous studies show that PHP systems were more vulnerable than applications written in ASP.NET and Java. By contrast, in 2015, 69% of Java applications suffered from vulnerabilities, while PHP systems were less vulnerable, 56% in 2015 compared to 76% in 2013.

Webserver configurations do not receive due attention.

The most common administrative errors were the same: information disclosure and insufficient protection against Brute Force. Apache Tomcat, WebLogic, Nginx, and IIS were vulnerable to these attacks.

The most vulnerable websites are banking and IT web applications.

All banking and IT websites contained critical vulnerabilities, results similar to 2014. There was improvement only in the manufacturing industry and telecoms applications.

Production sites are unprotected.

The percentage of vulnerable applications already put into production was extremely high: more than a half (63%) contained critical vulnerabilities. These vulnerabilities allow an attacker to obtain full control of the system (in case of arbitrary file upload or command execution) or sensitive information as a result of SQL Injection, XXE, etc. An intruder also can conduct a DoS attack. It is important to use application firewalls to protect production sites.

Analysis of web application source code is necessary.

The white-box testing revealed five times more critical vulnerabilities than black- and gray-box testing. This figure is two times higher for medium-level vulnerabilities. Automated tools for the source code analysis were highly efficient.

3. Participant Portrait

This study includes companies from a wide range of industries including manufacturing, banking, e-commerce, telecoms, IT, and governmental institutions.

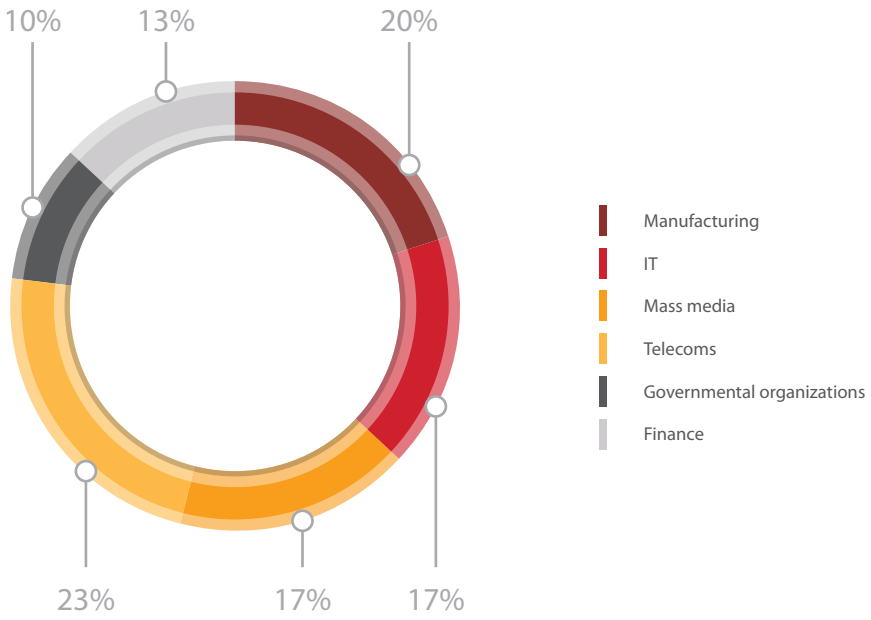

Figure 1. Systems by industry

The web resources analyzed were written in various programming languages with the use of different technologies. Among the systems studied, Java- and PHP-based web applications were the most common. Applications based on other languages and technologies, such as ASP.NET, Perl, ABAP, and 1C, were also used. The percentage of these applications was small in number in 2015 and they are grouped together in the category "Other".

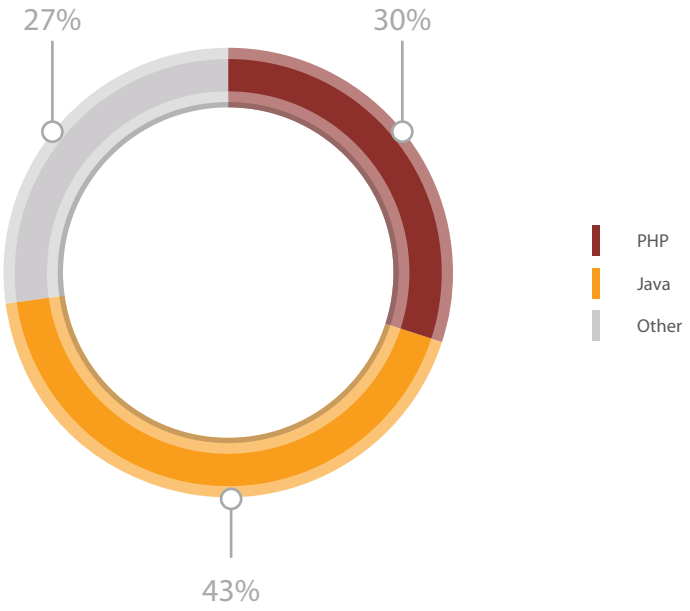

Figure 2. Systems by development technologies

Based on the web server used, all web applications studied were subdivided into four groups: run by Nginx, Microsoft Internet Information Services (IIS), Apache Tomcat, and web server WebLogic. Applications run by Apache and SAP NetWeaver Application Server were studied as well. These applications were included in the category "Other". The most widely used web server in 2015 was Nginx: 34% of web applications were based on it.

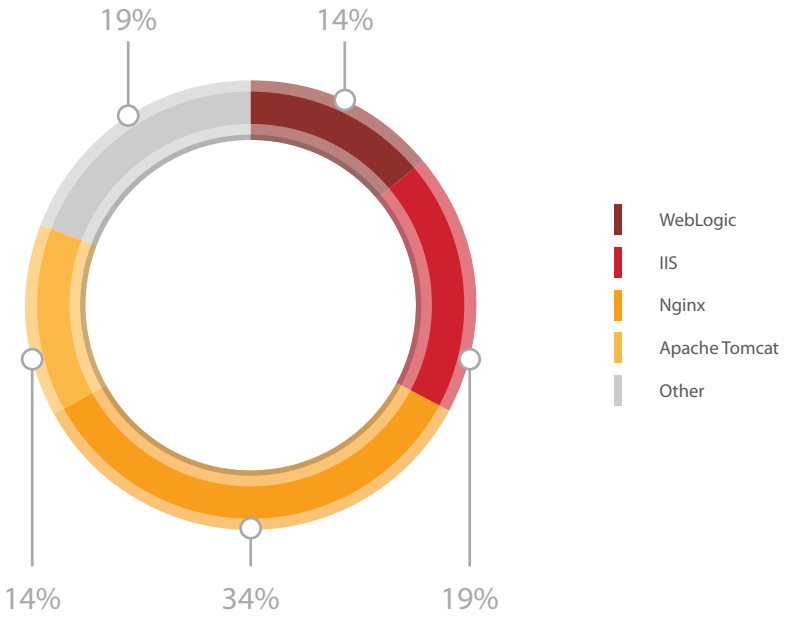

Figure 3. Systems by web servers used

Web applications studied include both production systems available over the Internet and testbeds under development or acceptance for operation. More than half (53%) of the resources studied were production systems, but the percentage of applications still in development was also high.

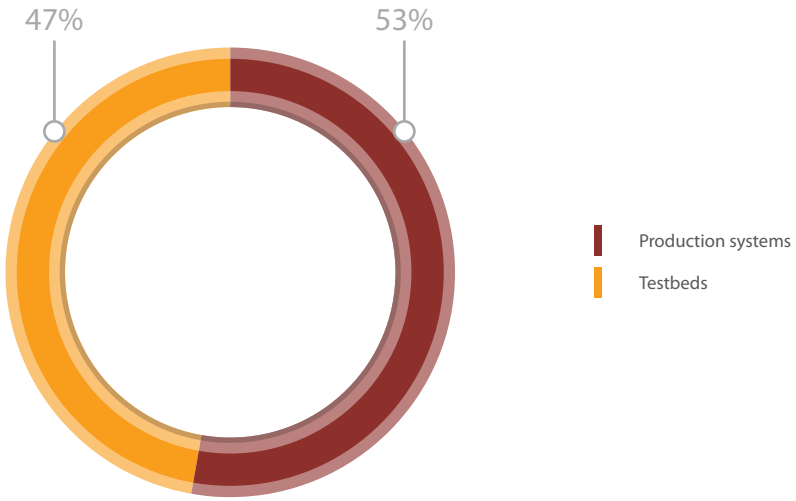

Figure 4. Systems by environment

As in 2013 and 2014, 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.

4. Vulnerability Statistics

This section includes analysis of the frequency and severity of various vulnerabilities classified according to the threats represented in WASC TC v. 2.

4.1. Most Common Vulnerabilities

According to our research in 2015, one in three web applications (34%) contained a critical vulnerability. Most of the vulnerabilities detected (62%) were classified as medium severity.

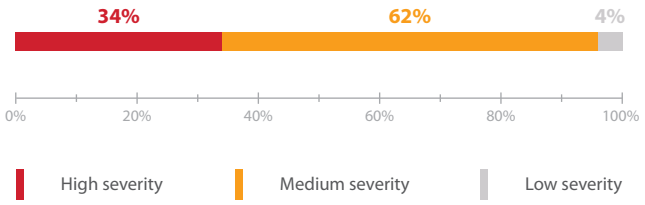

Figure 5. Vulnerabilities by severity

All web applications contained at least medium-severity vulnerabilities. According to the statistics, the percentage of applications with critical vulnerabilities is growing. As web technologies become more and more popular, the application functionality becomes more complex and thus hard to implement. As a result, developers find it difficult to detect and fix errors when developing systems that may make these systems expose to critical vulnerabilities.

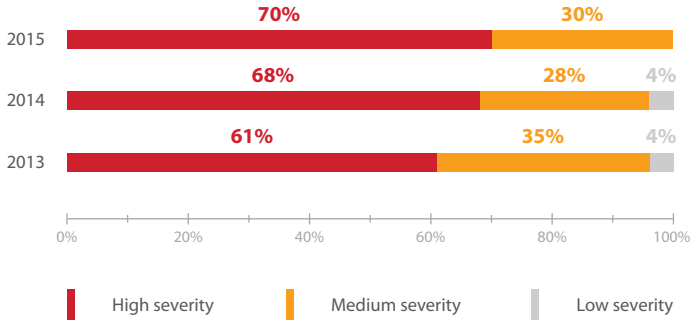

Figure 6. Websites by maximum severity

70% of the systems studied had a critical vulnerability, and the percentage of systems with this type of vulnerability has grown consistently over the last three years. The percentage of applications with medium-severity vulnerabilities was extremely high in 2015. The percentage of systems with low-severity vulnerabilities decreased from 80% to 50%. This might be caused by the easy detection and mitigation of low-severity vulnerabilities: you do not need to change application code significantly (for example, the percent of the Fingerprinting vulnerability dropped by two times).

80% of the investigated resources were vulnerable to the Cross-Site Scripting (XSS) attacks. In 2014, this vulnerability was only the second most common. Successful exploitation of the vulnerability could allow an attacker to inject arbitrary HTML tags, including JavaScript, into a browser, obtain a session ID or conduct phishing attacks.

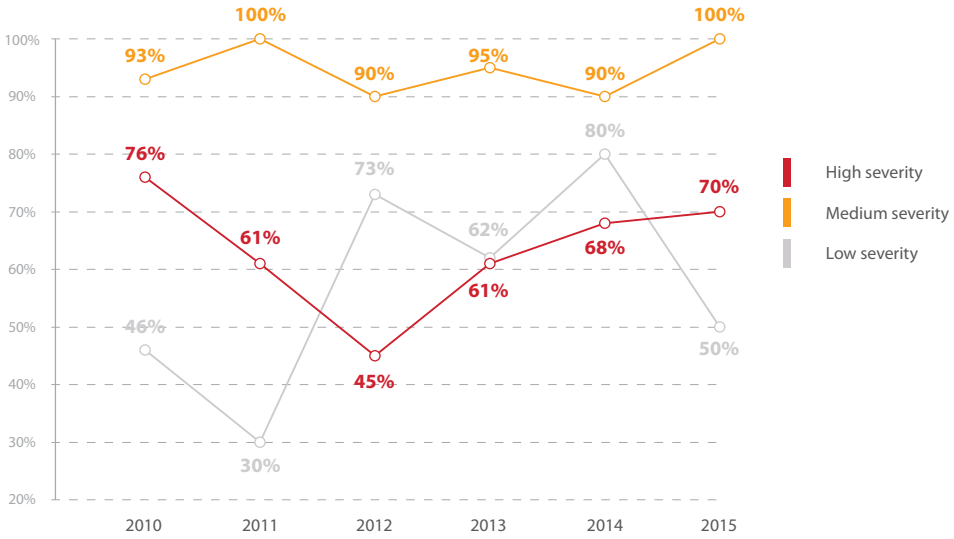

Figure 7. Websites by vulnerability severity

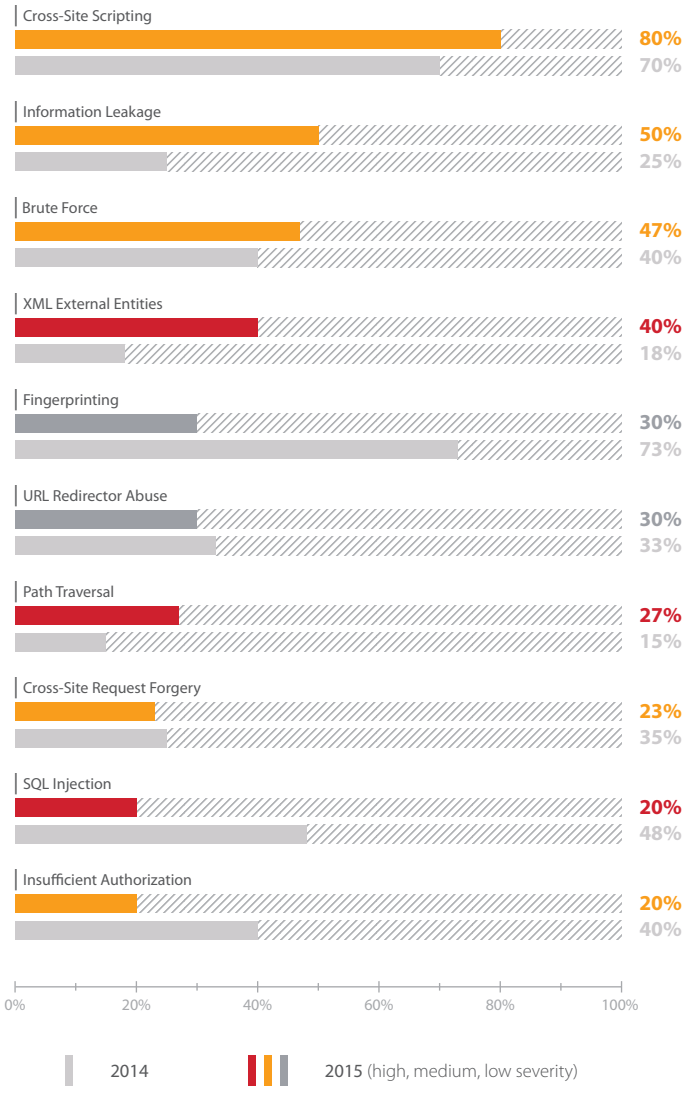

Figure 8. Most common vulnerabilities

The second most common flaw was Information Leakage: about 50% of the applications were vulnerable. The following sensitive information can be disclosed and used by an intruder for attacks: source code fragments, personal and account data, configuration server information. 47% of the websites were exposed to brute-force attacks.

The ten most common vulnerabilities also included SQL Injection, but in 2015 it was only the ninth most common. The XML External Entities vulnerability were among the most common high-severity vulnerabilities discovered in 2015. This vulnerability was the fourth most common. This security weakness allows attackers to obtain the content of server files or execute requests in the local network of the attacked server. The application does not check data received from a user properly allowing an attacker to change request logic via DTD Schema injection into an XML document. It allows an attacker to obtain sensitive information, source codes, configuration files and conduct a DoS attack.

The percentage of applications vulnerable to Fingerprinting, the most common flaw in 2014, decreased significantly in 2015.

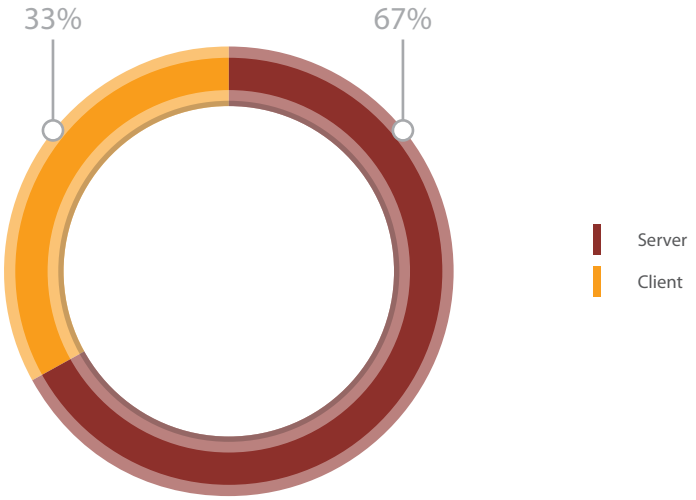

Figure 9. Distribution of vulnerabilities by attack targets

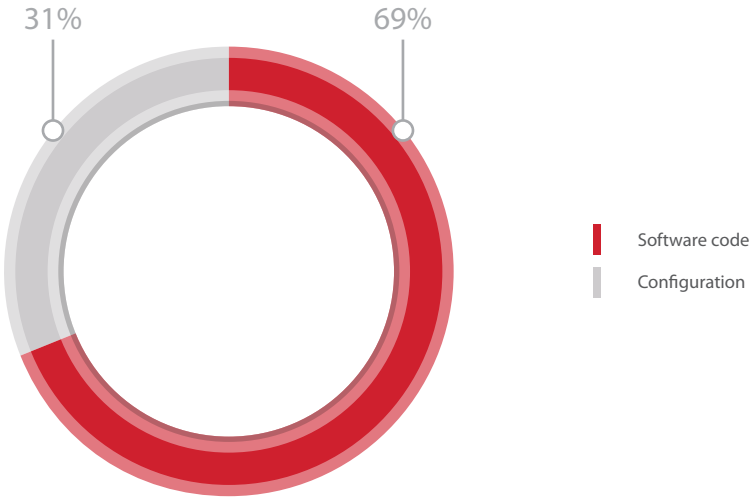

Figure 10. Distribution of vulnerabilities by source

Most of the vulnerabilities (67%) were detected on the sever components of web applications. Information Leakage and Brute Force are the most common examples of the vulnerabilities. A number of weaknesses that allow unauthorized individuals to attack client applications has decreased in half. Cross-Site Scripting (XSS), URL Redirector Abuse, and Cross-Site Request Forgery are among these flaws.

Similarly to previous years, a majority of vulnerabilities (69%) were found in software code, while 31% resulted from web application misconfiguration.

4.2. Vulnerabilities in Web Apps by Development Tools

More than a half of the sites written in PHP (56%) contained critical vulnerabilities. While the same value for the web resources written in Java was higher and made up 69%. High-severity vulnerabilities of the applications developed with other tools together totalled 88%.

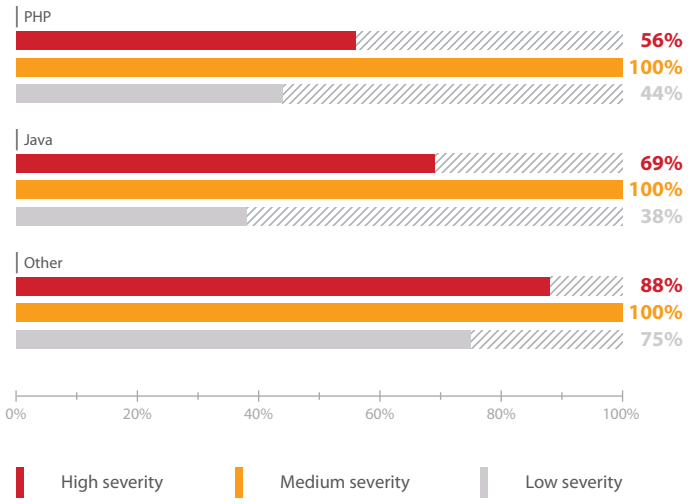

Figure 11. Systems with vulnerabilities of various severity levels (by development tools)

All resources included in the study were found to have vulnerabilities of at least a medium-severity level.

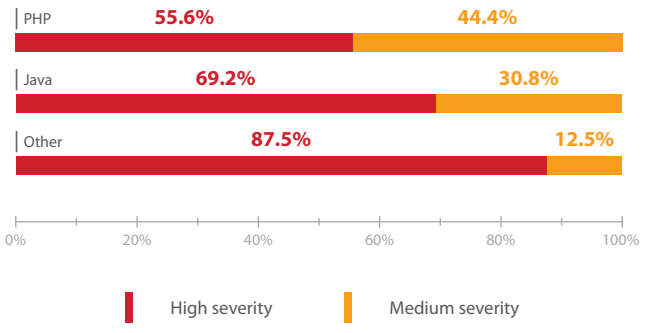

Figure 12. Systems by maximum vulnerability severity

An average PHP application contains 9.1 critical vulnerabilities, while in 2014, there were 11 vulnerabilities on average. A Java application contains 10.5 critical vulnerabilities, while applications based on other languages and development tools have only 2 vulnerabilities per application on average.

On average, every application contains more than 10 medium-severity vulnerabilities. In 2014, this figure for a PHP application was two times higher.

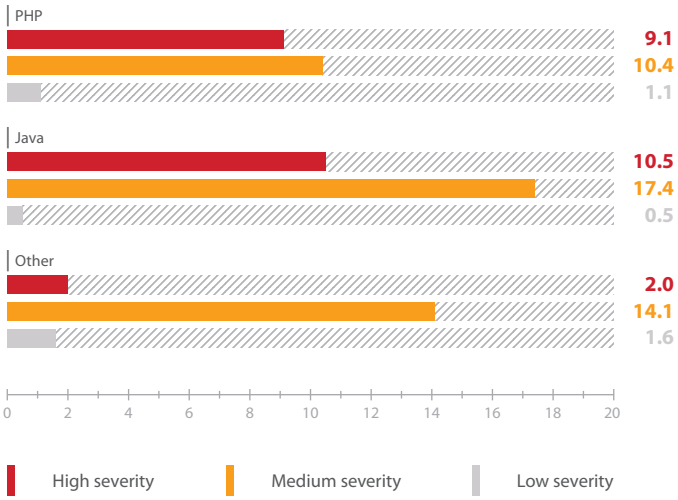

Figure 13. Average number of vulnerabilities per system by development tools

The table includes statistics on the frequency of common vulnerabilities among resources developed by different tools.

Table 1. Most common vulnerabilities

PHP	% of websites	Java	% of websites	Other	% of websites
Cross-Site Scripting	89%	Cross-Site Scripting	77%	Cross-Site Scripting	75%
Information Leakage	56%	XML External Entities	54%	Information Leakage	75%
Brute Force	33%	Brute Force	46%	Brute Force	63%
OS Commanding	22%	Path Traversal	31%	Fingerprinting	60%
SQL Injection	22%	Information Leakage	31%	XML External Entities	50%
Path Traversal	22%	URL Redirector Abuse	31%	Cross-Site Request Forgery	38%
Insufficient Authorization	22%	SQL Injection	23%	Insufficient Transport Layer Protection	38%
Fingerprinting	22%	Cross-Site Request Forgery	23%	URL Redirector Abuse	38%
URL Redirector Abuse	22%	Application Misconfiguration	23%	Path Traversal	25%
XML External Entities	11%	HTTP Response Splitting	23%	Insufficient Authorization	25%

The XXS vulnerability appeared the most widely spread (75%-89%) among all the types of programming languages. XML External Entities and Path Traversal took position in the top 10 most common vulnerabilities of web resources.

The percentage of SQL Injection found in the PHP applications in 2015 decreased from 67% to 22%. 23% of the Java applications were vulnerable to SQL Injection.

The figures below show how web resources developed by different tools are exposed to vulnerabilities most widespread in 2015.

Each of the most common vulnerabilities was detected in the resources studied.

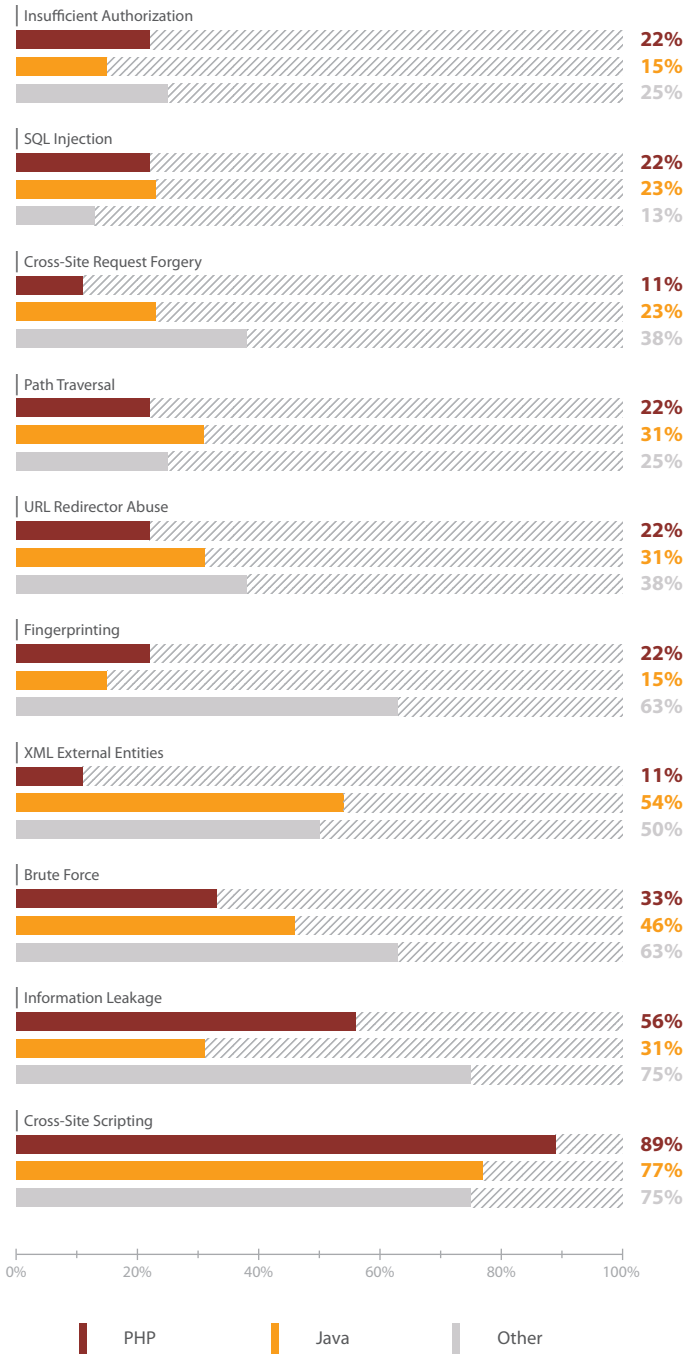

Figure 14. Applications with the most widespread vulnerabilities by development tools

4.3. Misconfigurations in Different Web Servers

The percentage of applications run on Nginx with high-severity vulnerabilities decreased in 2015 from 86% to 57%, the same value was in 2013. The percentage of applications run on Microsoft IIS with high-severity vulnerabilities increased significantly. A number of vulnerabilities in Apache Tomcat sites decreased to 33%. WebLogic-run applications were analyzed by the first time and 67% of them contained critical vulnerabilities.

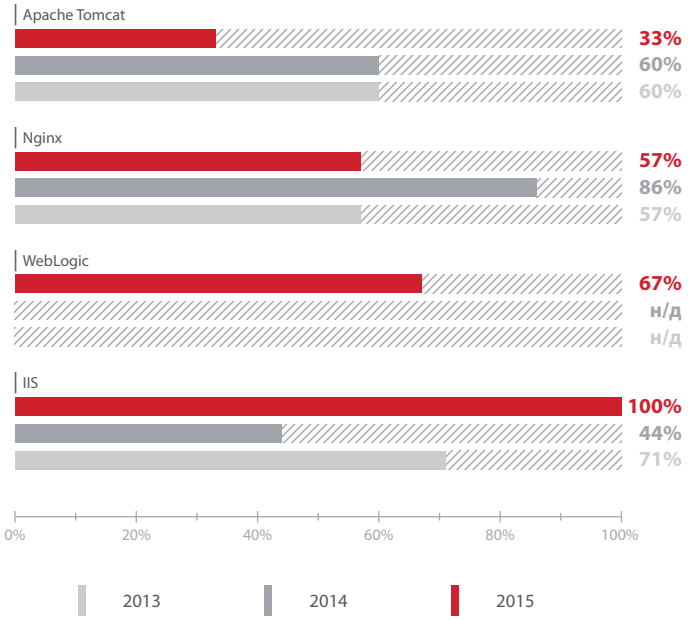

Figure 15. Web applications with high-severity vulnerabilities (by web servers)

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.

Table 2. Rating of administration vulnerabilities for different web servers

Nginx	% of websites	IIS	% of websites	Apache Tomcat	% of websites	WebLogic	% of websites
Brute Force	57%	Information Leakage	100%	Information Leakage	67%	Information Leakage	67%
Information Leakage	43%	Brute Force	75%	Brute Force	33%	Brute Force	33%
Fingerprinting	43%	Fingerprinting	75%	Fingerprinting	33%	Insufficient Transport Layer Protection	33%
Insufficient Transport Layer Protection	14%	Insufficient Transport Layer Protection	50%			Server Misconfiguration	33%
Server Misconfiguration	14%	Server Misconfiguration	25%			Application Misconfiguration	33%
		Application Misconfiguration	25%				

The most common administrative error was Information Leakage. This weakness was detected in all applications based on Microsoft IIS. On all the web servers tested, the most common administration flaw was Brute Force, present in the majority of Nginx-based resources. Fingerprinting was only the third most common and was not found in applications run on WebLogic.

4.4. Statistics by a Variety of Industries

All banking and IT websites contained critical vulnerabilities, results similar to 2014. Bank-owned applications were the most vulnerable in 2014. Mass media had 80% of applications with critical vulnerabilities; the same value as in 2013.

Almost a half of web applications used in the manufacturing industry and in telecoms were found to have critical vulnerabilities in 2015. However, there was improvement in comparison to the 2014 results.

As in 2013, the least vulnerable (33%) web applications belong to government organizations. It is worth noting that these results are not a fair representation of the government sector as a whole, since testing was mostly confined to a few projects where security awareness was very high. On the whole, the security of government web resources is worse than testing shows, as it is uncommon for governmental institutions to analyze the security of their web applications. It would be helpful to implement web application firewall in case of absence of regular security assessment.

In 2013 and 2014, testing of some industry applications confined to a few projects, therefore, comparison of its results are excluded.

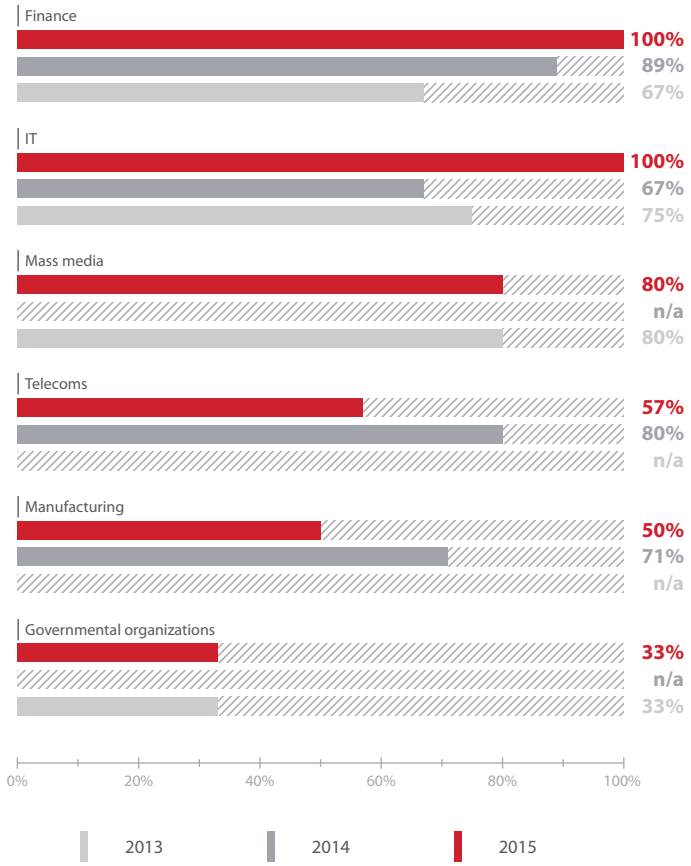

Figure 16. Sites with high-severity vulnerabilities by industries

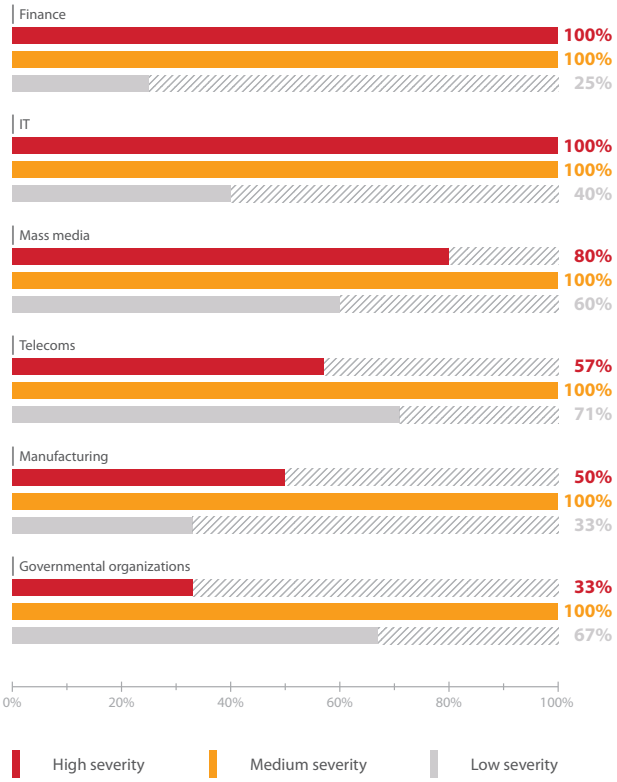

Figure 17. Sites exposed to vulnerabilities of a specific severity level for each industry considered

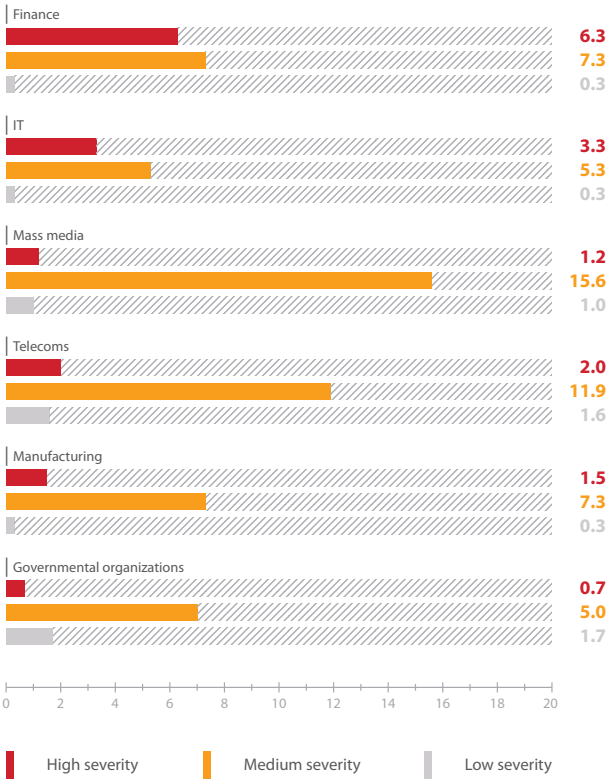

Figure 18. Average number of vulnerabilities per system by industries

Based on the average number of vulnerabilities per system, the least protected sites with three critical flaws per application belong to financial organizations and IT companies. The web resources of banks and financial institutions had 6.3 critical vulnerabilities. Within the mass media and manufacturing industry, each application on average had 2 and 1.5 critical flaws respectively. The data can be explained by the fact that these systems were analyzed via the white-box testing, including with the aid of the automated source code analyzer.

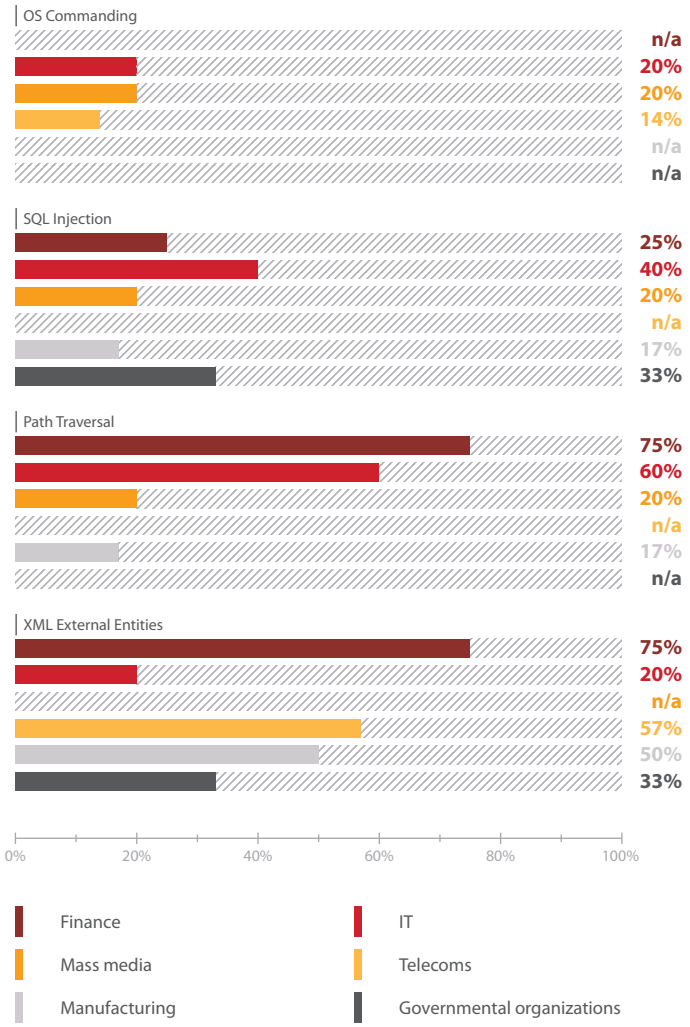

Figure 19. Vulnerable sites in a variety of industries

The most common critical vulnerabilities in 2015 were XML External Entities, SQL Injection, OS Commanding, and Path Traversal. However, there were no applications with all four flaws together.

The research results show that the majority of the systems in almost all of the industries tested are exposed to high-severity vulnerabilities. All the systems in the IT and finance sectors were vulnerable.

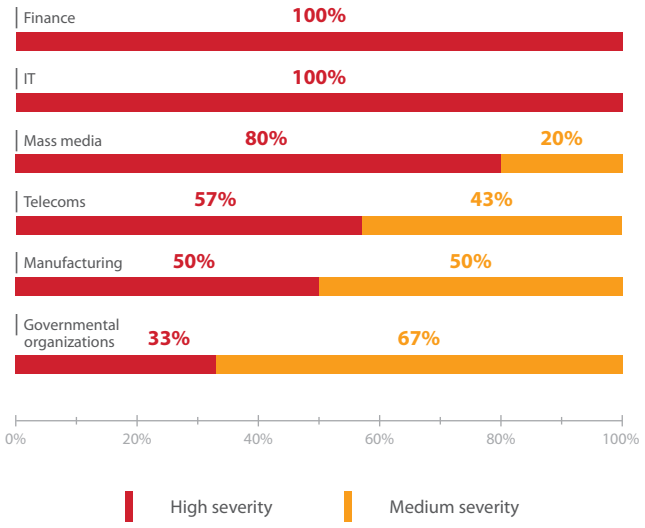

Figure 20. Web applications by maximum vulnerability severity

4.5. Test Technique Comparative Analysis

In 2015, the majority of results were obtained during gray- and black-box testing. The source code analysis was applied to 40% of the projects studied, and in 30% of cases, it was an automated analysis.

Positive Technologies experts carried out a comparative analysis of black-, gray-, and white-box testing techniques. It is worth noting that in one project the source code analysis detected dozens of SQL Injection, OS Commanding, Path Traversal and more than a hundred vulnerabilities in application configurations. The gray-box testing in another project uncovered dozens of the Cross-Site Scripting (XSS) vulnerabilities. These results do not represent correctly the reality and distort average statistical data, therefore, they were omitted.

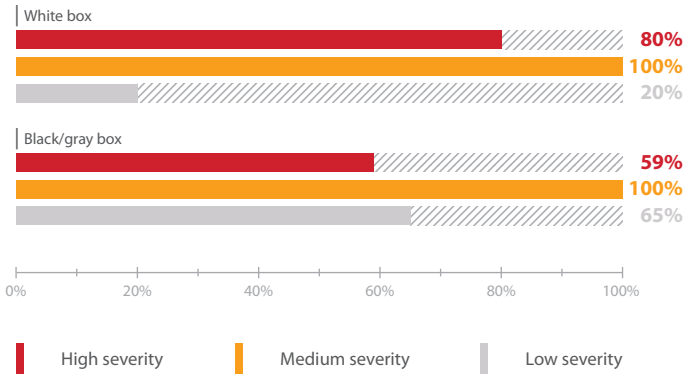

Figure 21. Systems with vulnerabilities of various severity levels (by testing techniques)

All web resources contained medium-severity vulnerabilities. The source code analysis uncovered more high-severity vulnerabilities than the black-box technique; however, even black- and gray-box testing discovered a high percentage of critical flaws (59%). Even if an intruder does not have access to source code, web applications are not necessarily secure; very often

extra knowledge of the system is not required to detect and exploit high- and medium-severity vulnerabilities. We obtained the same results in 2014.

The average number of different severity vulnerabilities detected by the white-box testing was higher than the results that came from black- and gray-box testing. Comparing the average number of vulnerabilities found per system shows that the white-box testing detects more high- and medium- severity vulnerabilities. Moreover, the amount will increase when we add the results of the projects omitted. Efficiency analysis of manual and automated (using the source code analyzer) testing proves the above-mentioned statement (see section 4.5.1).

There is almost no difference in numbers of XXS detected by two different testing methods. The source code analysis was more effective in detecting OS Commanding and XML External Entities.

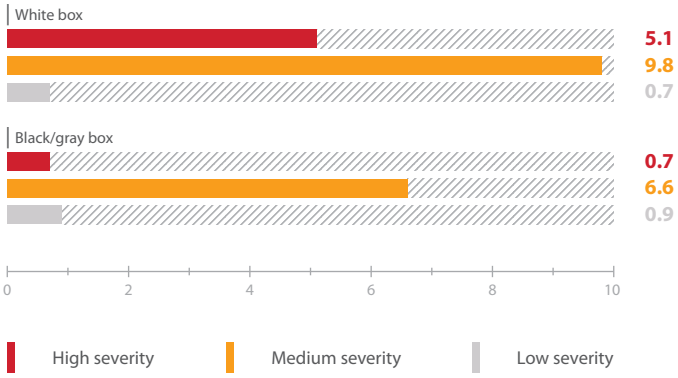

Figure 22. Average number of vulnerabilities per system by testing techniques

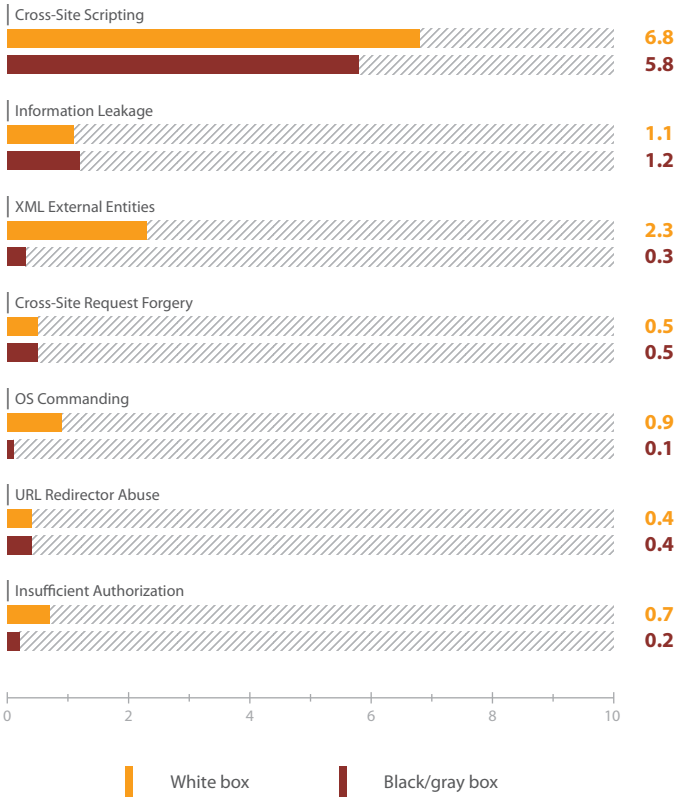

Figure 23. Average number of certain vulnerabilities per system by testing techniques

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 action accelerating DoS attacks. Therefore, the white-box testing is highly efficient. Statistical results argue for the source code analysis conducted alongside other testing techniques for better detectability.

4.5.1. Efficiency Analysis of Manual and Automated Testing Methods

The study includes the analysis of manual and automated (using automated source code scanners) white-box testing. As these techniques were implemented to different web resources, the data obtained as the result of analysis is average and cannot be used for comparison of these two methods.

Almost 40% of detected vulnerabilities are critical, the rest of them are medium-severity. This is explained by the fact that the analyzer classified vulnerabilities in accordance with the developer's requirements, ignoring the WASC TC v. 2 classification. Some low-severity vulnerabilities can be classified as Application Misconfiguration, that is a medium-severity flaw.

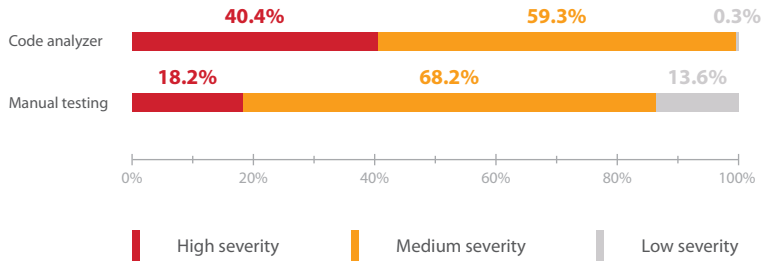

Figure 24. Vulnerabilities of various severity levels

Code analyzer discovered on average 15 critical and 20 medium-severity vulnerabilities per system. The above mentioned difference in the classification explains why the manual testing uncovered more low-severity vulnerabilities.

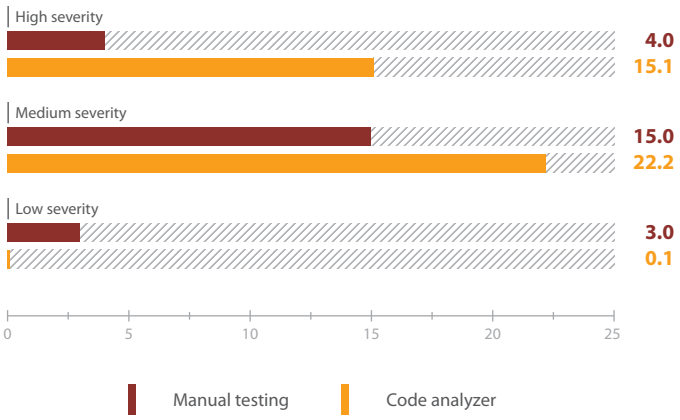

Figure 25. Average number of specified severity vulnerabilities per system

Diagrams below show the top list of vulnerabilities, detected by manual and automated testing.

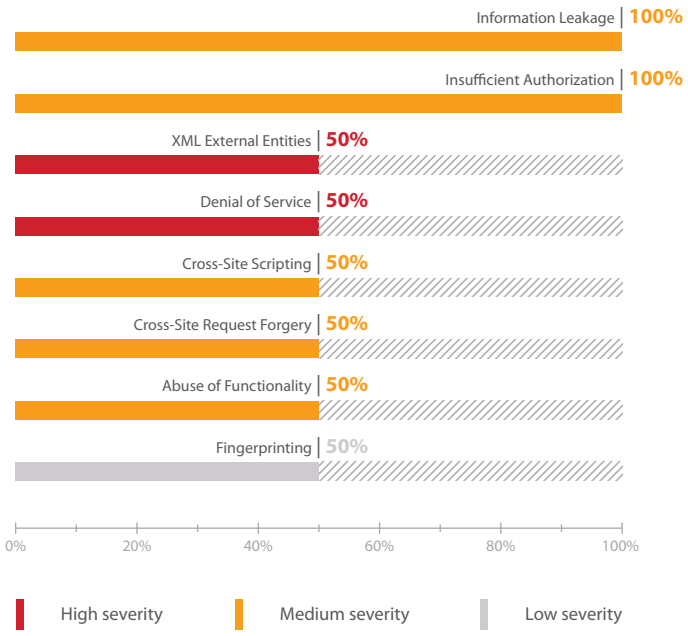

Figure 26. Vulnerabilities detected by the manual testing

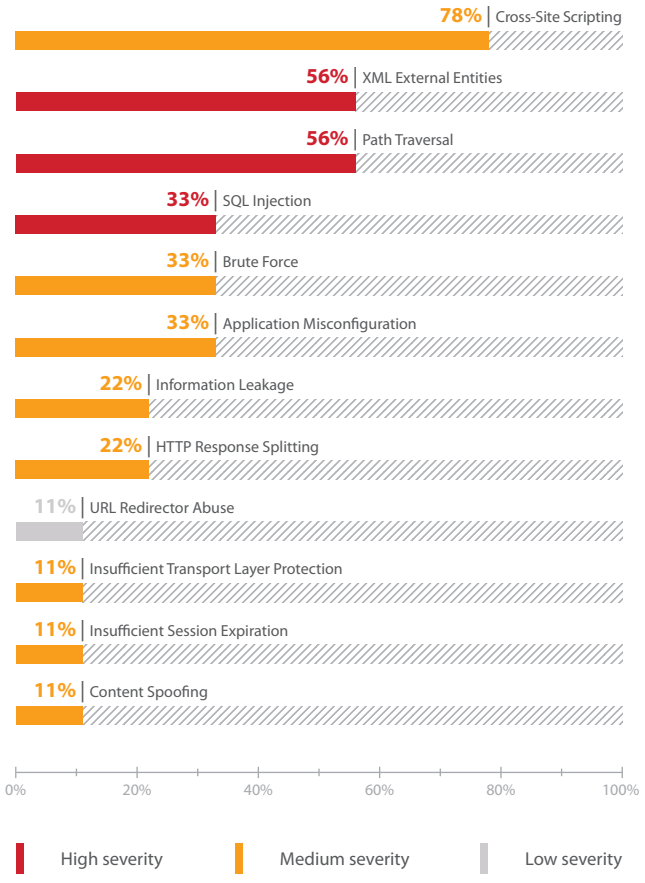

Figure 27. Vulnerabilities detected by source code analyzer

Thus, the white-box testing (both manual and automated methods) is more efficient than other methods without source code analysis.

It will take a few days or even weeks to investigate code of applications with numerous libraries using manual testing techniques. This method also requires a team of specialists to eliminate human errors (one specialist may overlook some flaws). Automated source analysis completes this task in a few hours or days regardless of the system complexity. The automated testing with source code analysis allows testers to:

+ Conduct testing faster
+ Exclude human errors
+ Execute testing at any development stage without buying security analysis services

4.6. Vulnerabilities in Test and Production Applications

This section provides the results of the security analysis of test and production systems. The figure below demonstrates the percentage of different severity vulnerabilities in already deployed systems and testbeds. According to the results, the systems still in development had more critical vulnerabilities.

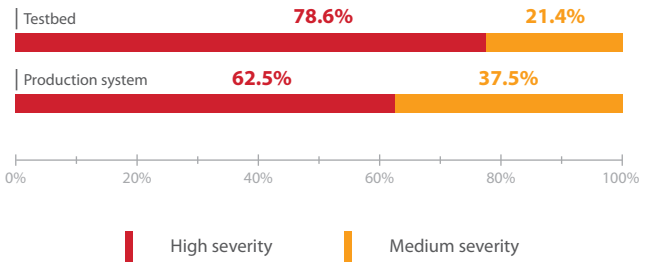

Figure 28. Systems by maximum vulnerability severity

62.5% of the production web resources had a critical vulnerability. The production sites had many flaws that could be exploited by a hacker. We obtained the same results in 2014.

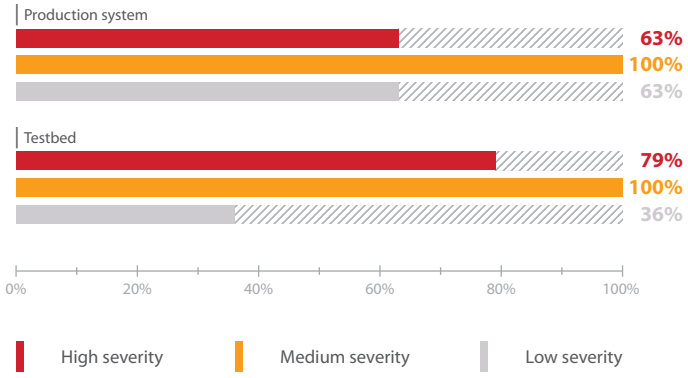

Figure 29. Vulnerable sites in test and production systems

The average number of high-severity vulnerabilities detected in the test systems is 10.6, almost twice as many as in the production applications (5.4). The white-box testing detected the majority of critical vulnerabilities. It is worth noting that the source code analysis was conducted mostly for the test systems.

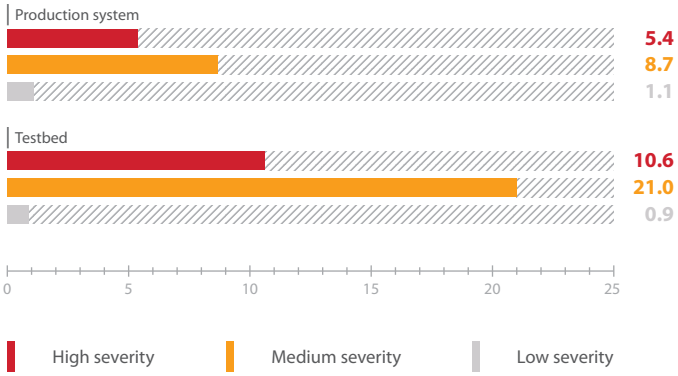

Figure 30. Average number of vulnerabilities per application in test and production systems

These results confirm that production systems had a low protection level.

It is important to regularly analyze test and production system security. These vulnerabilities in systems already in production indicate that secure software development lifecycle processes (secure SDLC) is not being followed. Secure application development will reduce the number of weaknesses available to attackers. It is important to use application firewalls to protect production sites.

Summary

This research argues that the general security level of web applications is decreasing. Code developers pay more attention to the functionality than to the security of web applications. As a result, many administration flaws appeared, they are classified as low-severity vulnerabilities, but by exploiting them, a hacker can obtain sensitive data (in case of its disclosure on the pages of application) or gain unauthorized access (as a result of brute-force or session attacks).

All applications regardless of development tools and industry are vulnerable. Each application studied had at least a medium-severity vulnerability, while 70% had critical flaws. An intruder may exploit code errors not only to obtain full control over the server, but also to attack application users that may cause significant reputational damages.

The 2015 results demonstrate a necessity to regularly analyze web applications security. It is important to analyze security at all development stages and regularly (e.g. twice a year) in the course of operational use. According to the results, the white-box testing with source code analysis is more efficient than other methods. Moreover, this testing can be conducted automatically (by the use of the source code analyzer) that is more usable than manual testing.

More than a half (62.5%) of applications put into production contained critical vulnerabilities. This can lead to sensitive data disclosure, system compromise or failure. It is important to use preventive protection measures — Web Application Firewall.

To reduce risks related to application vulnerabilities, vendors should ensure the usage of all above listed protection measures.

About Positive Technologies

Positive Technologies is a leading global provider of enterprise security solutions that include vulnerability and compliance management, incident and threat analysis, and application protection solutions. Our commitment to clients and research has earned Positive Technologies a reputation as one of the foremost authorities on Industrial Control Systems (ICS), Banking, Telecom, Web Application, and ERP security. Recognition from the analyst community, including Gartner, IDC, and Forrester have seen us described as '#1 fastest growing* company in vulnerability management' and 'most visionary in web application protection**'. To learn more about Positive Technologies please visit 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+.

**Source: Gartner Magic Quadrant for Web Application Firewalls 15 July 2015.

POSITIVE TECHNOLOGIES

pt@ptsecurity.com ptsecurity.com