SECURITY TRENDS & VULNERABILITIES REVIEW WEB APPLICATIONS

2017

POSITIVE TECHNOLOGIES

Contents

Introduction 3
1. Materials and methods 3
2. Executive summary 4
3. Participant portrait 5
4. Trends 6
5. Manual web application security testing 7
    5.1. Most common vulnerabilities 8
    5.2. Analysis of threats and security levels 10
    5.3. Statistics by a variety of industries 13
    5.4. Vulnerabilities in web applications by development tools 15
    5.5. Vulnerabilities in test and production applications 18
    5.6. Test technique comparative analysis 19
6. Automated security assessment 20
Conclusions 24

INTRODUCTION

Every year, web applications expand their presence in more and more areas. Almost every business has its own web applications for clients and for internal business processes. However, application functionality is often prioritized at the expense of security, which negatively affects the security level of the entire business.

As a result, web application vulnerabilities provide massive opportunities for malicious actors. By taking advantage of mistakes in application architecture and administration, attackers can obtain sensitive information, interfere with web application functioning, perform DoS attacks, attack application users, penetrate a corporate LAN, and gain access to critical assets.

This report provides statistics gathered by Positive Technologies while performing web application security assessments throughout 2016. Data from 2014 and 2015 is provided for comparison purposes.

This information suggests paths of action: which security flaws in web applications require attention during development and operation, how to distinguish potential threats, and what the most effective techniques for security assessment are. We also illustrate trends over time in web application development in the context of information security.

1. MATERIALS AND METHODS

Data for this report is drawn from 73 web applications examined in 2016 for which Positive Technologies conducted in-depth analysis. Some of the applications are publicly available on the Internet, while others are used for internal business purposes. We excluded vulnerabilities detected in the course of penetration testing, perimeter scanning, and online banking security audits; this information can be found in the respective reports.[1]

Vulnerability assessment was conducted via manual black-, gray-, and white-box testing (with the aid of automated tools) or automated source code analysis. Black-box testing means looking at an application from the perspective of an external attacker who has no prior or inside knowledge of the application. Gray-box testing is similar to black-box testing, except that the attacker is defined as a user who has some privileges in the web application. The most rigorous method, white-box scanning, presupposes the use of all relevant information about the application, including its source code. Results of manual security assessment are given in Section 5 of this report, while the results of automated scanning are in Section 6 .

Vulnerabilities were categorized according to the 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 as part of a number of other attacks. In addition, we distinguished three categories of vulnerabilities: Insecure Session, Server-Side Request Forgery, and Clickjacking. These categories are absent from the WASC classification, but can be often found in the web applications studied.

The Insecure Session category includes session security flaws, such as missing Secure and HttpOnly flags, which allow attackers to intercept the user's cookies in various attacks.

Server-Side Request Forgery is a vulnerability that allows sending arbitrary HTTP requests while posing as the system. After receiving a URL or an HTTP message, a web application performs an insufficient destination check before sending a request. An attacker can exploit this vulnerability and send requests to servers with restricted access (for example, computers on a LAN), which can result in disclosure of confidential data, access to application source code, DoS attacks, and other problems. For example, an attacker can obtain information about the structure of network segments that are not available to external users, access local resources, and scan ports (services).

Clickjacking is a kind of attack on users involving visual deception. In essence, a vulnerable application is loaded in a frame on the application page and is disguised as a button or another element. By clicking this element, a user performs the attacker-chosen action in the context of that website. The vulnerability that makes this attack possible occurs when the application does not return an X-Frame-Options header and therefore allows showing the application in frames. In some browsers, this vulnerability also allows performing a Cross-Site Scripting attack.

1 www.ptsecurity.com/ww-en/analytics/

Our report includes only code and configuration vulnerabilities. Other widespread security weaknesses, such as flaws in the software update management process, were not considered.

The severity of vulnerabilities was calculated in accordance with the Common Vulnerability Scoring System (CVSS v. 3). Based on the CVSS score, our experts assigned vulnerabilities one of three severity levels: high, medium, or low.

2. EXECUTIVE SUMMARY

All web applications analyzed have vulnerabilities.
Security flaws were found in all the applications analyzed. 58 percent had at least one high-severity vulnerability. At the same time, we see a positive trend: the number of websites with high-severity vulnerabilities decreased by 12 percent compared with 2015.

Application users are not protected.
Most of applications allow attacks on their users. Moreover, a number of applications provide insufficient protection of user data. For instance, we gained access to personal data of 20 percent of the applications that process user information, including bank and government websites.

Leaks are still a pressing problem.
Approximately half of web applications are exposed to leaks of critical data, including source code and personal data. 63 percent of web applications disclose the versions of software in use.

Web application vulnerabilities are an easy vector for LAN penetration.
One in every four web applications allows attacks on LAN resources. For example, an attacker can access files, scan hardware on the LAN, or attack network resources. Besides, one out of every four applications was vulnerable to SQL Injection (high severity), which allows attackers to access the application database. In addition, this vulnerability could allow an attacker to read arbitrary files or create new ones, as well as launch DoS attacks.

Manufacturing companies are the most vulnerable.
Almost half of manufacturing web applications received the lowest grade possible. The majority of web applications in all industries—with the exception of finance—were exposed to high-severity vulnerabilities. In finance, "only" 38 percent of applications had high-severity vulnerabilities.

64 percent of ASP.NET applications contain high-severity vulnerabilities.
Additionally, approximately one out of every two PHP and Java applications contains high-severity vulnerabilities . PHP applications were particularly affected, with an average of 2.8 per application.

Production systems are more vulnerable than test applications.
In 2016, production systems turned out to be less protected. During manual testing, high-severity vulnerabilities were found on 50 percent of testbeds and on 55 percent of production systems. The number of high- and medium-severity vulnerabilities per application on production systems was twice as high compared to test systems.

Source code analysis is more effective than black-box testing.
Manual analysis of source code enabled our experts to detect high-severity vulnerabilities in 75 percent of applications. Black-box testing revealed such vulnerabilities on only 49 percent of web applications.

Automated testing is a fast way to find vulnerabilities.
Automated analysis of source code found an average of 4.6 high-severity, 66.9 medium-severity, and 45.9 low-severity vulnerabilities per application. Source-code analysis with the help of automated tools can identify all exit points—in other words, all possible exploits for each vulnerability—reliably and rapidly.

3. PARTICIPANT PORTRAIT

The applications represent companies from a wide array of industries, including finance, government, media, telecoms, manufacturing, and e-commerce.

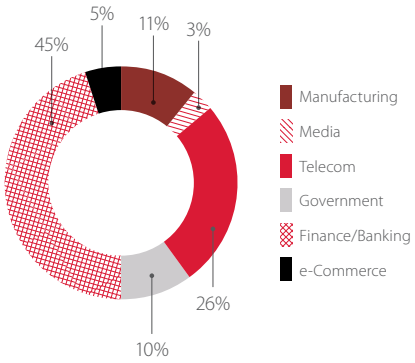

Figure 1. Participant portrait

Almost two-thirds of these applications (65%) were production sites (in other words, currently operating and available to users).

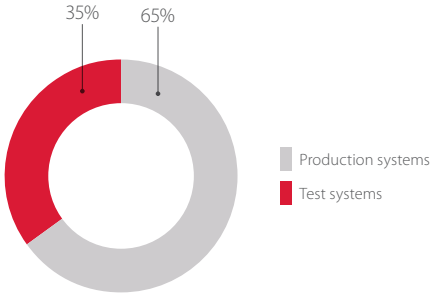

Figure 2. Production and test systems

This year, PHP and Java were the most common development languages used. The proportion of ASP.NET applications increased year-over-year. Development languages in the "Other" category (Ruby, Python, etc.) accounted for only 7 percent.

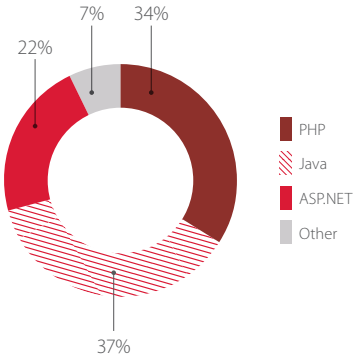

Figure 3. Web application development tools

4. TRENDS

All web applications, whether examined using manual or automated security assessment tools, contained vulnerabilities with various severity levels. Only 1 percent of applications had solely low-severity vulnerabilities. We can see some improvement in the percentage of applications with high-severity vulnerabilities, which fell from 70 percent in 2015 to 58 percent in 2016. This improvement is partially driven by the fact that companies took account of last year's security findings when developing new web applications, and, perhaps most importantly, concentrated on remediating high-severity vulnerabilities.

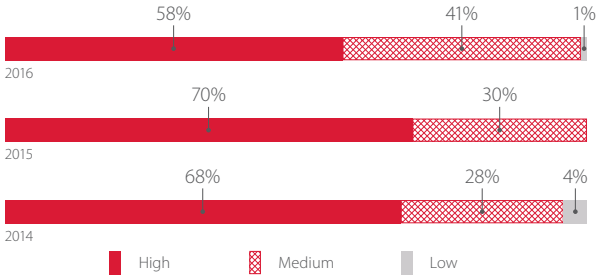

Figure 4. Percentage of web applications whose worst vulnerabilities were of high, medium, or low severity

In general, we observed a discouraging trend in high-severity vulnerabilities during the three previous research periods. But growth in these vulnerabilities slowed down in 2015, and finally in 2016 they actually fell. Still, critical flaws were found in more than half of applications.

Medium-severity vulnerabilities were detected in almost all applications. Every year, this percentage is consistently in the range of 90 to 100 percent. The percentage of web applications with low-severity vulnerabilities increased.

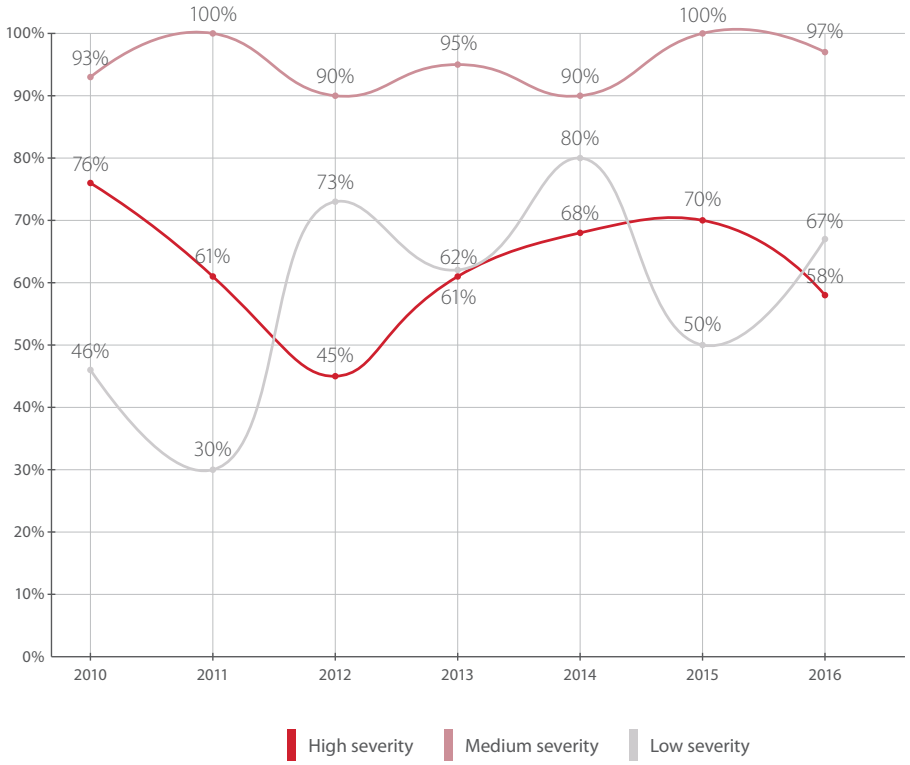

Figure 5. Websites by vulnerability severity

5. MANUAL WEB APPLICATION SECURITY TESTING

Out of all vulnerabilities detected by manual testing, the majority (81%) were of medium severity, with one tenth being of high severity. Compared to 2015, the share of high-severity vulnerabilities substantially decreased, but this is explained by the fact that in 2016 far more medium-severity vulnerabilities per application were detected.

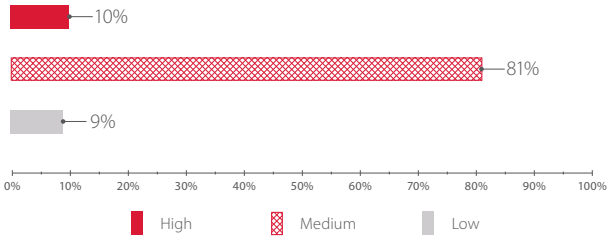

Figure 6. Vulnerabilities by severity (results of manual analysis)

Security flaws were found in all web applications. Manual testing uncovered high-severity vulnerabilities in more than half of analyzed applications (54%), 44 percent of applications contained medium- and low-severity vulnerabilities, and a mere 2 percent of applications had only low-severity vulnerabilities.

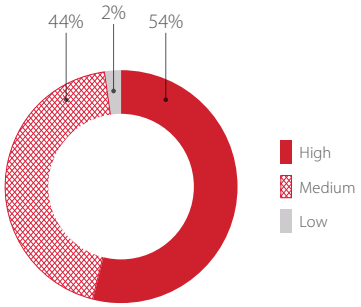

Figure 7. Web applications by maximum vulnerability severity (results of manual analysis)

On average, manual analysis found 17 medium-severity, 2 high-severity, and 2 low-severity vulnerabilities per application.

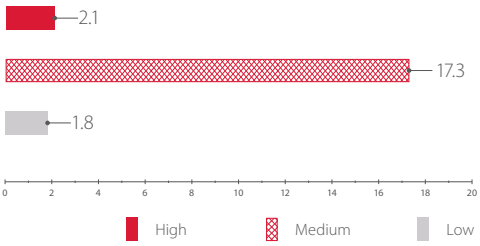

Figure 8. Average number of vulnerabilities per application (results of manual analysis)

5.1. MOST COMMON VULNERABILITIES

In 2016, half of the top 10 vulnerabilities allowed performing attacks against web application users.

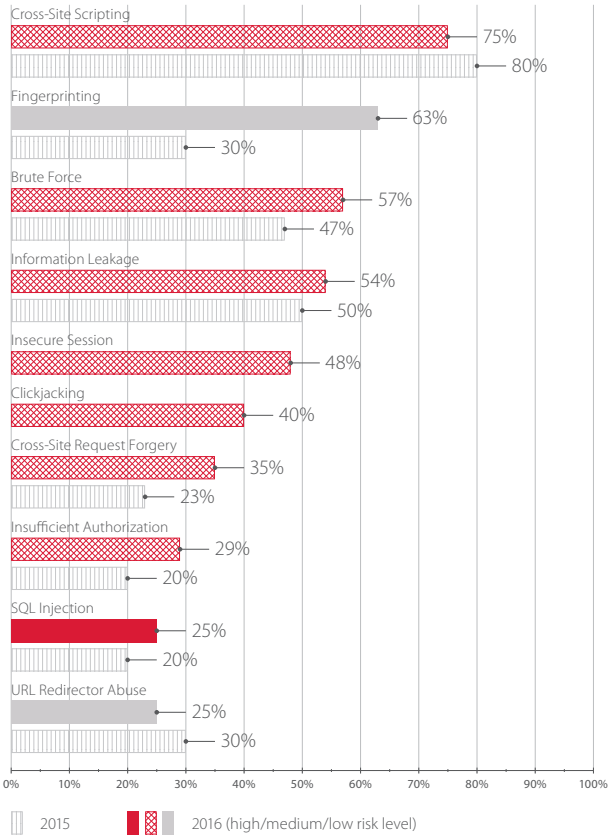

Figure 9. Most common vulnerabilities detected by manual testing (percentage of web applications)

As in 2015, Cross-Site Scripting (medium severity) tops the list and was found in 75 percent of the web applications examined. Successful exploitation of this vulnerability could allow an attacker to inject arbitrary HTML tags and JavaScript scripts into a browser session, obtain a session ID, conduct phishing attacks, and more.

Similarly to past years, Positive Technologies used its information on attacks against web applications to create a list of the most common attacks.[2] Sources of data are pilot projects involving deployment of PT Application Firewall. To hack a website or attack its users, attackers try to exploit various vulnerabilities in web application design and administration. Research revealed that 58 percent of applications that took part in pilot projects underwent attempts to attack users with Cross-Site Scripting—the most common vulnerability in this year's rating.

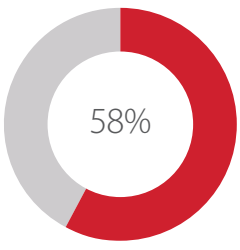

Figure 10. Cross-Site Scripting attack attempts (percentage of web applications)

2 https://www.ptsecurity.com/upload/corporate/ww-en/analytics/Web-Application-Vulnerability-2016-eng.pdf

Flaws leading to disclosure of information about the current software version (Fingerprinting) were found in 63 percent of applications, taking second place. In addition, more than half of web applications (54%) are vulnerable to Information Leakage, such as of source code and personal data.

Third place went to poor or non-existent protection against brute-force attacks. The percentage of applications vulnerable to this kind of vulnerability increased by 10 percent year-over-year.

Session security flaws and Clickjacking took the next two places in our top 10 list. These categories made their first appearance in 2016, so no comparison with the previous year is possible. While developers became more careful about eliminating high-severity vulnerabilities that threaten application owners, flaws causing damage to users took center stage this year. Vulnerabilities to Cross-Site Request Forgery, which also allows attacks on users, were detected in 35 percent of web applications.

As already mentioned, the total share of websites containing high-severity vulnerabilities has fallen, and only one high-severity vulnerability—SQL Injection—made the top 10 this year, yet still it was found in 25 percent of web applications. According to our research, this vulnerability was the most commonly exploited one in 2016: attackers attempted to exploit it in 84 percent of web applications.

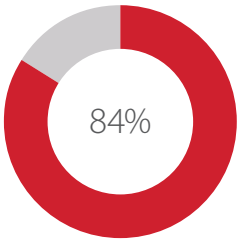

Figure 11. SQL Injection attack attempts (percentage of web applications)

Client-side vulnerabilities were detected in 59 percent of all applications examined in 2016. Among these vulnerabilities are Cross-Site Scripting, Cross-Site Request Forgery, session security flaws, and other security problems that make it possible to attack web application users. The remaining 41 percent of detected vulnerabilities, such as Information Leakage and Insufficient Authorization, are on the server side.

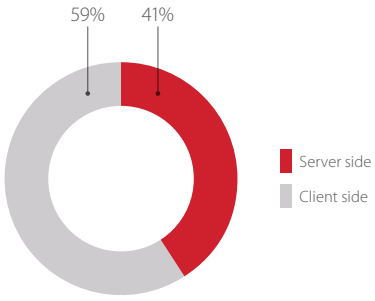

Figure 12. Vulnerabilities by attack target

Most of the vulnerabilities detected (73%) were found in software code and are connected with development mistakes—such as SQL Injection. Misconfiguration of web servers is responsible for about a quarter of all security flaws.

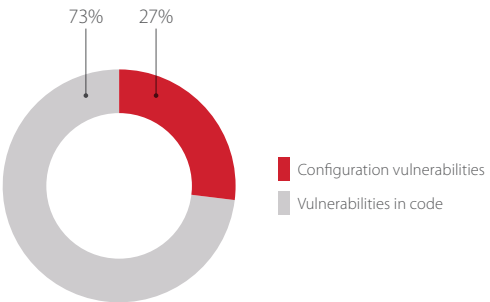

Figure 13. Vulnerability types

5.2. ANALYSIS OF THREATS AND SECURITY LEVELS

We graded web application security based on the possible consequences of exploitation of the vulnerabilities that we found, from "extremely poor" to "acceptable." An extremely poor security level means high-severity vulnerabilities that, for example, allow an external attacker to perform OS Commanding or lead to disclosure of sensitive information. In general, if a web application has vulnerabilities of high severity, its security level varies from "extremely poor" to "below average."

The overall level of web application security is still rather low. Experts rated the security of 16 percent of web applications as extremely poor.

One in every three examined web applications (32%) is characterized by a poor security level. Only 5 percent of applications are sufficiently protected.

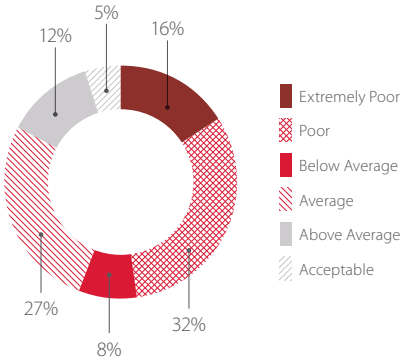

Figure 14. Web application security level

The lowest grades ("poor" and "extremely poor") in 2016 went to applications used by online stores, manufacturers, and telecommunications companies: more than half of them had poor or extremely poor security. More than a third of e-commerce (34%) and manufacturing (43%) web applications received the lowest grade possible, "extremely poor." The security of financial and governmental applications is marginally better. Only 15 percent of telecom web applications could boast of acceptable security. The sample size for media applications was insufficient for drawing conclusions.

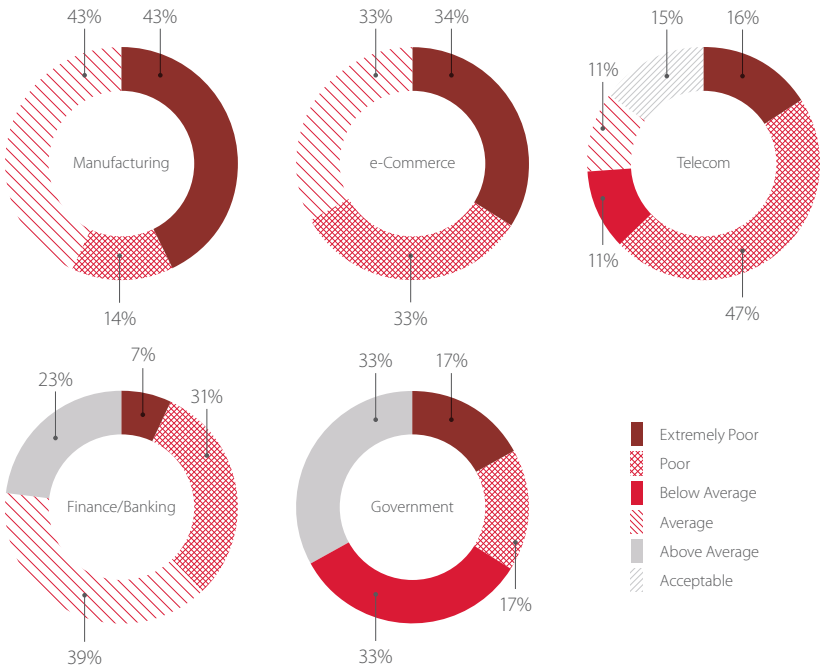

Figure 15. Web application security grade (by industry)

In 2016, the most widespread threat was attacks on web application users: such attacks are possible in nearly all web applications (94%). As mentioned in the previous section, a quarter of web applications contain vulnerabilities that can give an attacker access to databases. The same share of web applications (25%) can be a vector of penetration to a corporate LAN: these applications allow outsiders to scan hardware, learn about the network structure, and send requests to local nodes. About one in every five applications (19%) makes it possible to execute arbitrary OS commands on a server.

Note that DoS attacks were not attempted as part of application testing. Nevertheless, a number of applications had vulnerabilities that allow performing such attacks.

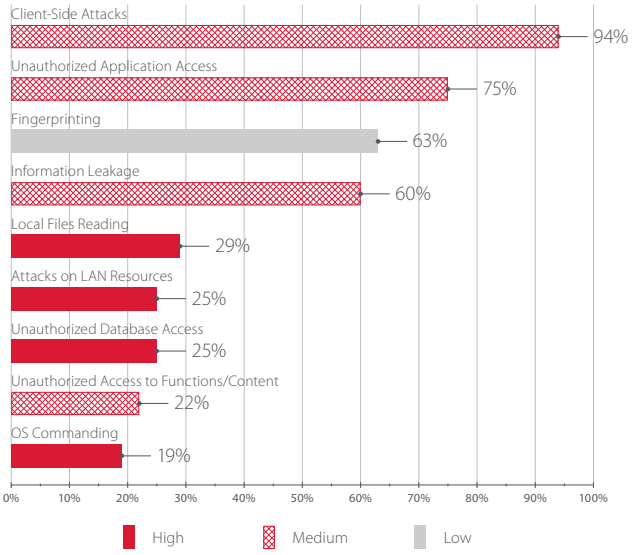

Figure 16. Most common threats

By and large, the vulnerabilities that enabled attacks on users were Cross-Site Scripting, Cross-Site Request Forgery, Open Redirect, Insecure Session, and Clickjacking. These development flaws are in this year's list of top 10 common vulnerabilities.

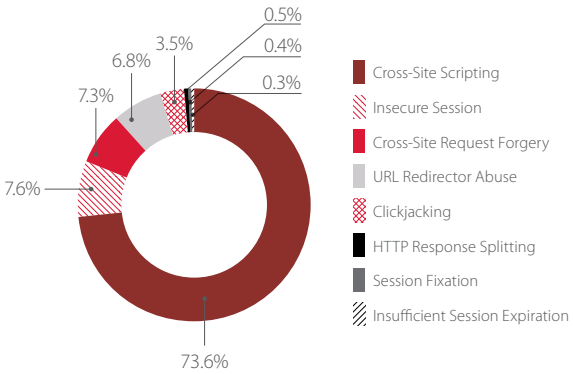

Figure 17. Vulnerabilities enabling attacks on users

An attacker can gain access to 70 percent of web applications. Such access is generally made possible by a weak password policy, absence of brute-force protection, and ability to conduct attacks on users.

The third and fourth places in our top 10 went to information leaks. Disclosure of information about the software version in use is a low-risk vulnerability, but in the case of outdated software, an attacker can take advantage of known vulnerabilities by using publicly available exploits.

By exploiting various vulnerabilities, we managed to obtain the source code of 8 percent of web applications. By analyzing source code, attackers can detect other vulnerabilities in a web application and advance an attack vector. Source code can contain sensitive information that enables access to critical resources.

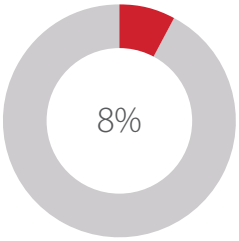

Figure 18. Percentage of web applications in which access to source code was possible

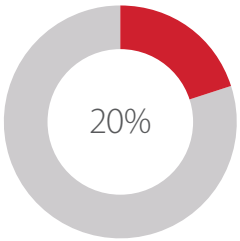

Figure 19. Percentage of web applications in which users' personal data can be obtained

Users' personal data is also under threat—attackers can gain access to 20 percent of web applications that process such data, including financial and governmental applications. Attackers can obtain information about users by taking advantage of an information leak or exploiting other vulnerabilities, such as SQL Injection.

While reviewing critical threats by industry, we can notice that governmental, financial, and telecom applications contain the full range of high-severity vulnerabilities. Access to DBMS and OS Commanding threats are more common among e-commerce and manufacturing web applications.

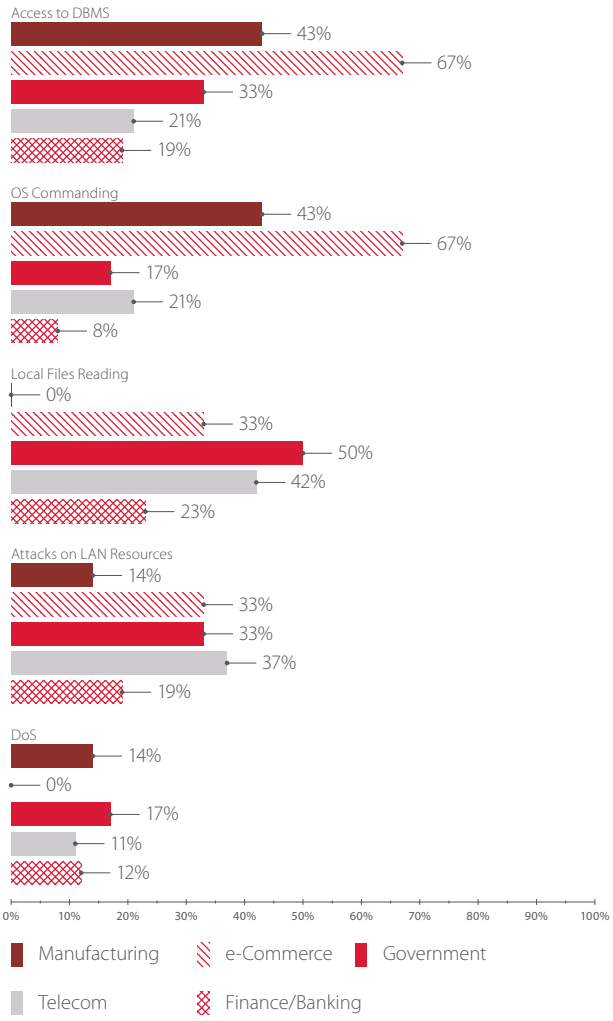

Figure 20. Critical threats by industry

5.3. STATISTICS BY INDUSTRY

This section provides per-industry statistics on telecom, financial, e-commerce, governmental, and manufacturing web applications. Statistics for media applications are not given here, since the sample size was insufficiently large.

The majority of web applications in all industries—with the notable exception of finance—were exposed to high-severity vulnerabilities. High-severity vulnerabilities were found on the sites of 74 percent of telecoms, 67 percent of governmental institutions and online stores, and 57 percent of manufacturing companies. In finance, only 38 percent of applications had high-severity vulnerabilities.

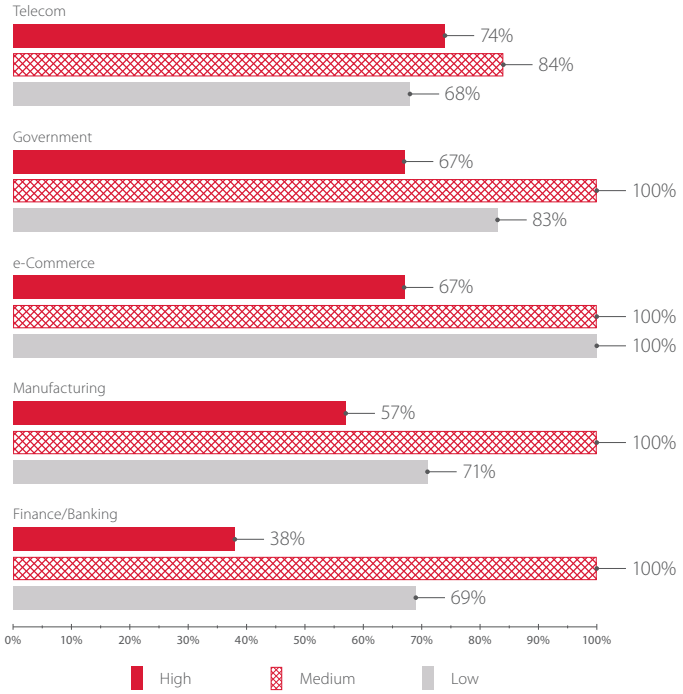

Figure 21. Web applications by vulnerability severity

Medium-severity vulnerabilities were found in all examined web applications, with the exception of some telecom applications. The telecom industry was prone to contrasts: many applications had high-severity vulnerabilities, but at the same time, there was a small contingent of relatively well-secured web applications containing only minor flaws.

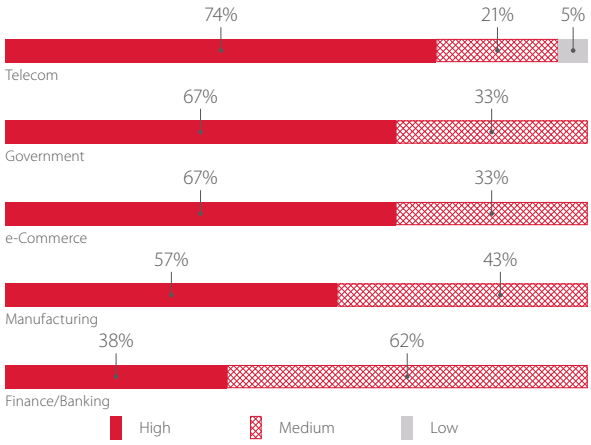

Figure 22. Web applications by maximum severity of vulnerabilities (percentage of applications)

Comparing web applications by their average number of vulnerabilities, governmental applications have more high-severity vulnerabilities than any other industry and rank first with 6.2 vulnerabilities per application. In 2015, this value was much smaller (0.7 vulnerabilities per web application). In previous years, security assessment was carried out only for important governmental web applications, for which security was one of the core development requirements. But now governments are broadening their attention to encompass existing web applications, including those with a rather low security level.

E-commerce web applications also have a large number of high-severity vulnerabilities. These web applications also have the highest rate of medium-severity vulnerabilities. On average, 39.3 vulnerabilities per application were detected in e-commerce applications, compared to 27.5 vulnerabilities in governmental applications.

About two high-severity vulnerabilities on average can be found per manufacturing or telecom application. The most secure are financial web applications, with only 0.8 high-severity vulnerabilities per application.

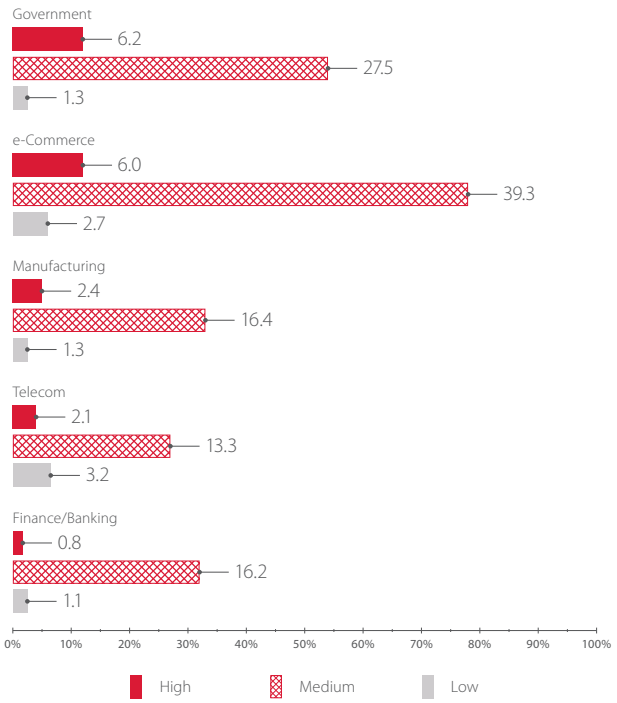

Figure 23. Average number of vulnerabilities per application by industry

In 2016, one of the most common high-risk vulnerabilities in web applications across all industries was SQL Injection. Other common vulnerabilities were XML External Entities, OS Commanding, and Path Traversal. Various telecom and financial applications contained all these flaws, but this may not necessarily be representative, since these industries also accounted for the majority of the dataset.

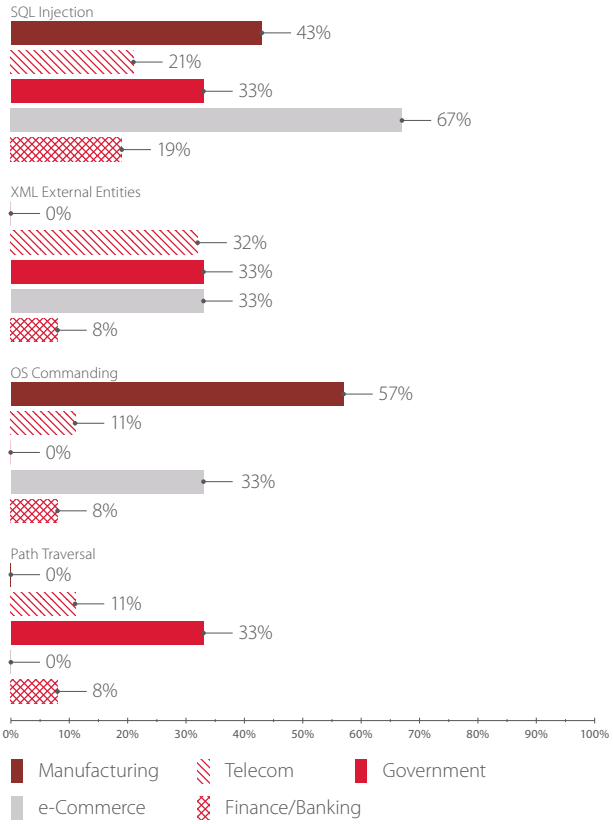

Figure 24. Percentage of websites with common vulnerabilities, by industry

5.4. VULNERABILITIES IN WEB APPLICATIONS BY DEVELOPMENT TOOLS

As in 2015, all examined applications, regardless of the development tool used, contained at least medium-level vulnerabilities. Statistics are given for PHP, Java, and ASP.NET applications. Applications written in other, less common languages were too few to provide meaningful statistics. However, almost all applications written in less common languages contained high-severity vulnerabilities, and in just one case were there only low-severity vulnerabilities.

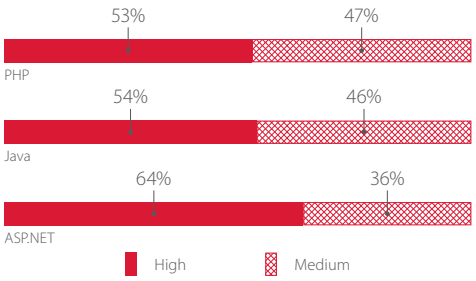

Figure 25. Web applications by maximum vulnerability severity

The choice of PHP versus Java for development had virtually no effect on the severity of application vulnerabilities in 2016. All applications had medium-severity vulnerabilities, and more than half contained high-severity vulnerabilities.

The highest rate of high-severity vulnerabilities was observed among ASP.NET applications, with 64 percent containing such vulnerabilities. As compared to PHP and Java, the percentage of ASP.NET applications with medium-severity and low-severity vulnerabilities is slightly lower: 93 and 50 percent respectively. However, the average ASP.NET application had fewer high-severity vulnerabilities than its PHP and Java counterparts.

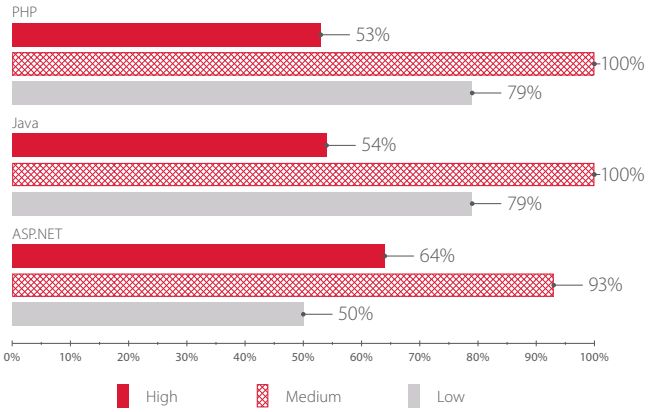

Figure 26. Web applications with vulnerabilities of various severity levels

As mentioned previously, the number of high-severity vulnerabilities has fallen significantly compared to previous years. On average, we see about two high-severity vulnerabilities per application, with PHP applications having the highest number of such vulnerabilities (2.8). The number of medium-severity vulnerabilities, on the contrary, increased compared to 2015 (for PHP and Java), with Java applications containing twice as many vulnerabilities of this level as other applications.

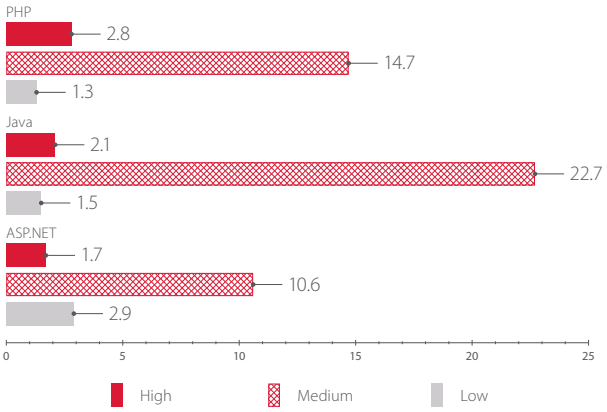

Figure 27. Average number of vulnerabilities per application, by development tools

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

PHP	% of websites
Cross-Site Scripting	79%
Information Leakage	79%
Brute Force	74%
Fingerprinting	74%
Insecure Session	53%
Clickjacking	42%
Insufficient Authorization	32%
SQL Injection	26%
OS Commanding	26%
URL Redirector Abuse	26%

ASP.NET	% of websites
Cross-Site Scripting	79%
Fingerprinting	79%
Brute Force	75%
Information Leakage	50%
Clickjacking	50%
Cross-Site Request Forgery	42%
Insecure Session	42%
Insufficient Authorization	33%
SQL Injection	29%
XML External Entities	21%

Java	% of websites
Cross-Site Scripting	64%
Insecure Session	57%
Cross-Site Request Forgery	50%
URL Redirector Abuse	43%
Deserialization of Untrusted Data	29%
Information Leakage	29%
SQL Injection	29%
Clickjacking	21%
Insufficient Authorization	21%
XML External Entities	14%

Severity levels: High Medium Low

Table 1. Most common vulnerabilities by development platform

The most common vulnerability in all applications was Cross-Site Scripting. More than 60 percent of applications, across all programming languages, were vulnerable to it. Security flaws related to Information Disclosure are also common: Information Leakage and Fingerprinting.

Compared to 2015, PHP and Java applications had fewer high-severity vulnerabilities. For example, Path Traversal, which was common in previous years, did not make the top 10 this year.

Nevertheless, 26 to 29 percent of applications in each category are vulnerable to SQL Injection, more than a quarter of PHP applications (26%) are vulnerable to OS Commanding, and one of the most common vulnerabilities for all other development tools is XML External Entities.

Regardless of development tool or language, however, applications across the board were generally exposed to common vulnerabilities, as shown in the following figures.

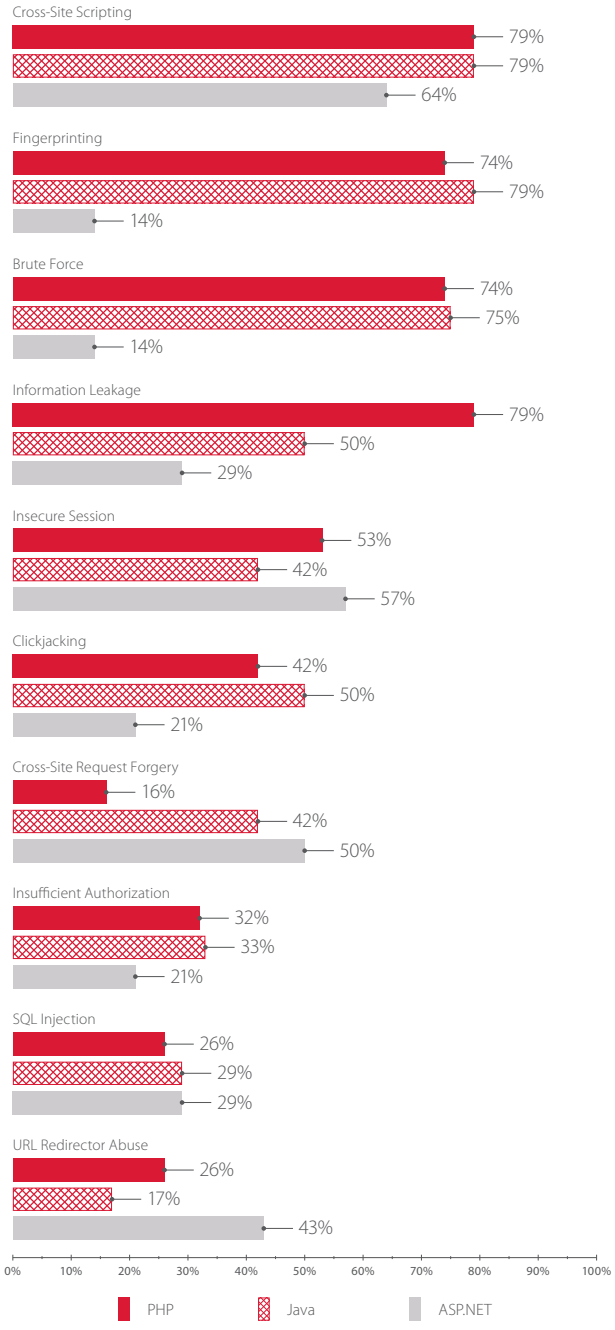

Figure 28. Applications with the most common vulnerabilities, by development tool

5.5. VULNERABILITIES IN TEST AND PRODUCTION APPLICATIONS

In 2016, production systems proved more vulnerable than test systems. High-severity vulnerabilities were found on 50 percent of testbeds and on 55 percent of production systems.

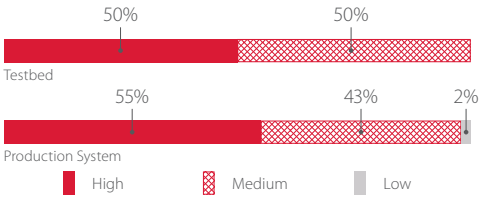

Figure 29. Systems by maximum vulnerability severity (percentage of systems)

Moreover, the number of high- and medium-severity vulnerabilities per application on production systems was twice as high compared to test systems. One explanation is that security-conscious companies (which test applications at the development stage, among other things) are better at avoiding vulnerabilities. Another reason is that deployment and implementation are complicated processes that add complexity, and therefore can cause new flaws. Some vulnerabilities can be detected only on a fully configured and ready-to-use system.

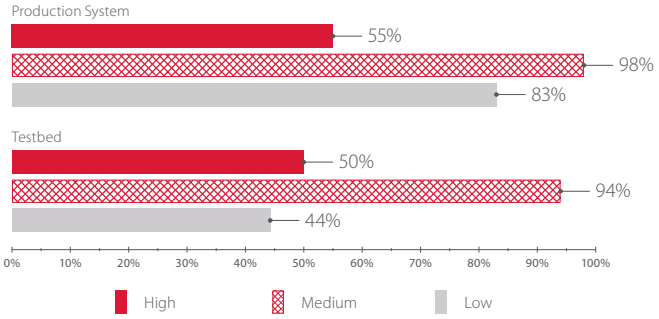

Figure 30. Web applications with various vulnerability severity levels

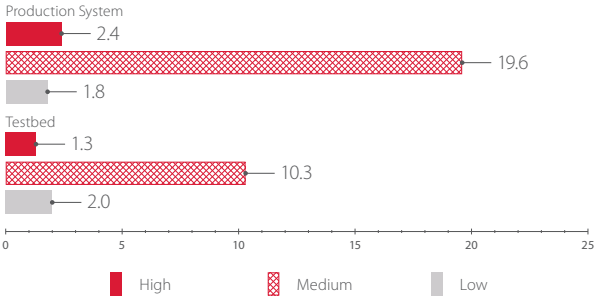

Figure 31. Average number of vulnerabilities per system

These results demonstrate the need to implement application security processes throughout the entire software lifecycle—from design and development to deployment and operation.

5.6. COMPARISON OF TEST TECHNIQUES: BLACK BOX, GRAY BOX, AND WHITE BOX

Manual analysis of security enabled our experts to apply black-, gray-, and white-box testing methods. Head-to-head comparison of these methods is impossible, since different methods were applied to different web applications, but we can look at the results to make a general assessment of test technique effectiveness. Most web applications (81%) were analyzed using black- and gray-box testing, without any access to the source code.

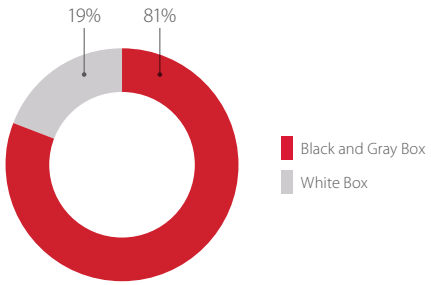

Figure 32. Applications by test technique used

White-box testing, with analysis of source code, enabled our experts to detect high-severity vulnerabilities in 75 percent of applications. Black-box testing, meaning that our testers did not receive source code or other key information, revealed such vulnerabilities on only 49 percent of web applications. Medium-severity vulnerabilities were found practically in all applications: 98 percent by black-box testing, and 92 percent by analyzing source code. Thus, white-box analysis proved to be more effective in most cases. However, even an attacker who does not have prior information about a web application still has a good chance of finding a high-severity vulnerability.

In addition, attackers can obtain source code by exploiting various vulnerabilities, as already demonstrated above.

Also note that in black- and gray-box testing, the testers were careful to avoid impacting application performance or causing denial of service. But real attackers are unlikely to be so considerate.

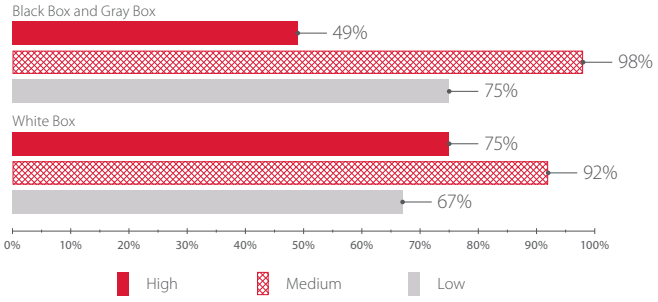

Figure 33. Percentage of web applications with vulnerabilities of a given category (by testing technique)

On average, Positive Technologies detected 2.8 high-severity vulnerabilities per application when analyzing source code (white box), and 1.9 vulnerabilities per application without source code (black box). The difference between these two figures is not so dramatic compared to the previous period, because many of the same companies had learned valuable lessons from the prior year's testing. White-box testing is highly effective, as can be confirmed by the automated testing results provided in the following section.

Moreover, white-box testing detected three times more low-severity vulnerabilities than black-box testing. There was a minimal difference in the number of medium-severity vulnerabilities detected using the various techniques.

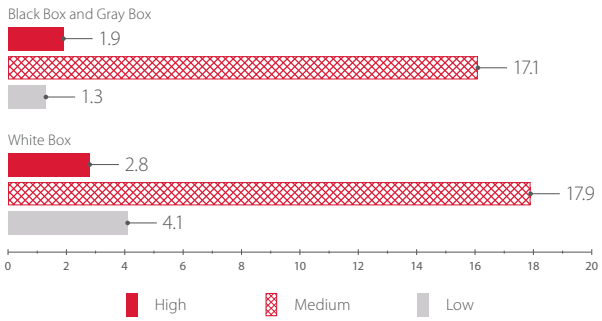

Figure 34. Average number of vulnerabilities per application (by testing technique)

As in previous years, white-box testing detected more high-severity vulnerabilities. For example, analysis of source code revealed XML External Entity issues four times more often than black-box testing did. Session security flaws, Cross-Site Request Forgery, and Open Redirect were detected primarily with the help of white-box testing.

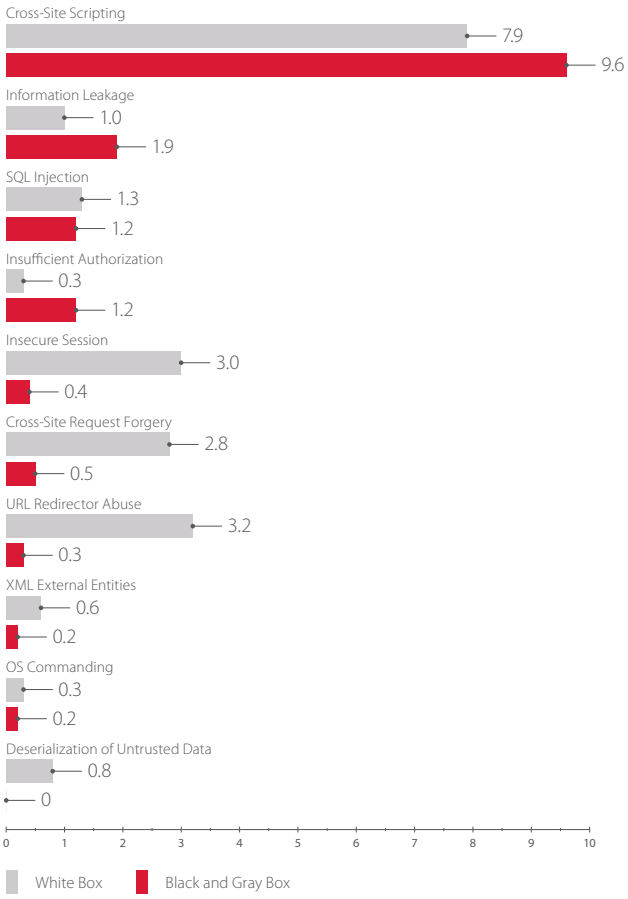

Figure 35. Average number of certain vulnerabilities per application by testing technique

6. AUTOMATED SECURITY ASSESSMENT

This section considers web applications that were subjected to analysis using an automated source code analyzer. Since manual and automated techniques were used on different web applications, same-application comparison of the results is not possible. Instead, we can consider the averages of the results obtained by these two different methods.

All applications analyzed here were pre-production, and some of them were at an early development stage. The vulnerabilities detected by automated scanning and represented in the statistics were confirmed manually using testbeds.

The vulnerability classification given here is the same used in the automated security scanner. This classification is different from the WASC classification thanks to its more detailed breakdown of weaknesses, which in the WASC classification are combined into general categories, such as Application Misconfiguration and Improper Filesystem Permissions.

We observe some improvements compared to the 2015 results. A quarter of vulnerabilities were of high severity (28.5%), compared to 40.4 percent in the previous year. Similar to the situation with manual checks in the previous section, part of this change may be due to the fact that the prior year's testing inspired companies to be more careful with security during development.

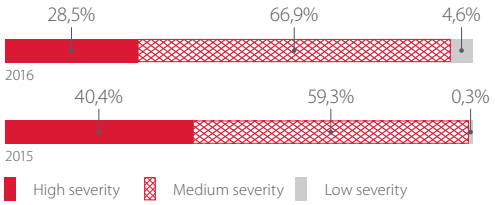

Figure 36. Severity of vulnerabilities found (automated testing)

All examined applications had at least medium-severity vulnerabilities. As in 2015, high-severity vulnerabilities were found in the vast majority of applications (89%).

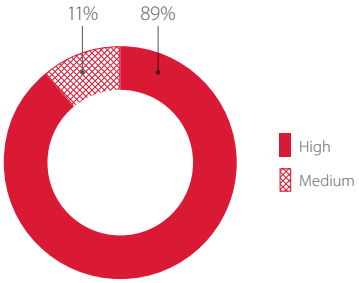

Figure 37. Web application distribution by maximum severity level (automated testing)

Medium-severity vulnerabilities were discovered in all examined web applications.

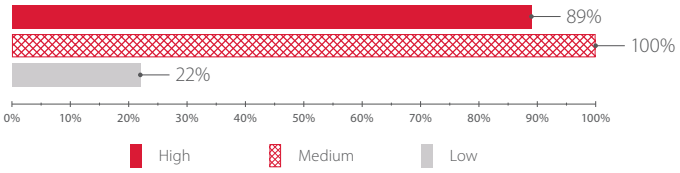

Figure 38. Web applications with vulnerabilities of various severity levels (automated testing)

Automated analysis of source code found an average of 4.6 high-severity, 66.9 medium-severity, and 45.9 low-severity vulnerabilities per application. Moreover, two of the examined applications had hundreds of high-severity vulnerabilities and around 2,000 medium-severity vulnerabilities, but these applications have been omitted here to prevent distortion of the results.At the same time, these values give an idea of the usability and effectiveness of automated analysis tools for improving the security of web applications. Source-code analysis, unlike black-box testing, can identify all exit points—in other words, all possible exploits for each vulnerability. This information is needed to ensure total elimination of vulnerabilities.

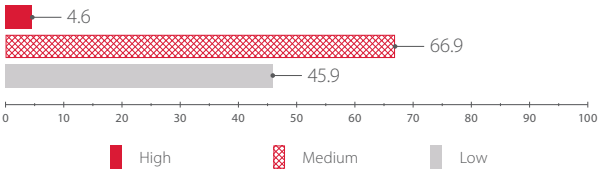

Figure 39. Average number of vulnerabilities found per application, by severity (results of automated analysis)

All web applications examined were vulnerable to Cross-Site Scripting. As manual testing confirms, this is the most common vulnerability in the applications we tested. Detection of one such vulnerability is shown in the following screenshot. The web application does not check user-supplied data, which allows attackers to target users by passing JavaScript commands, for example.

Figure 40. Example of Cross-Site Scripting detection

The code analyzer we used for testing can verify the vulnerabilities it finds by automatically generating exploits. In this example, the exploit was designed to send a request using the GET method.

The most common high-severity vulnerabilities were related to improper restrictions on file access. Almost half of the web applications allow creating and modifying arbitrary files, which enables execution of OS commands, such as if an attacker creates a PHP file. In almost all of these applications, such flaws exist together with Arbitrary File Reading/Removal vulnerabilities. An example of one such vulnerability in source code is shown in the following screenshot. The vulnerability allows an attacker to perform Path Traversal attacks and read arbitrary files on the server.

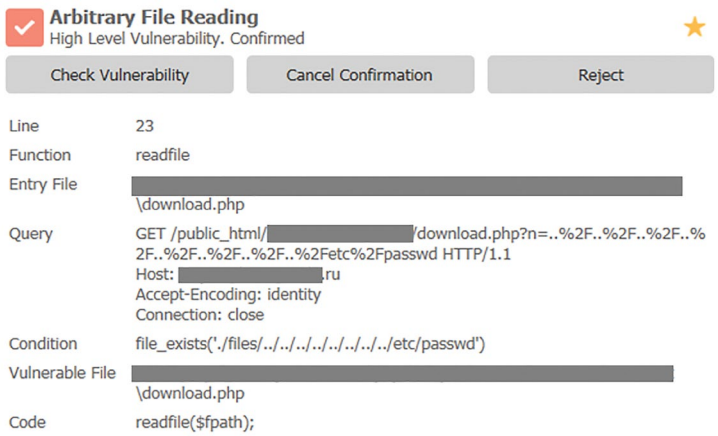

Figure 41. Example of Arbitrary File Reading detection

The source code of a number of web applications had a high-severity SQL Injection vulnerability related to insufficient input filtering. This vulnerability allows attackers to retrieve information from the database. In some cases, an attacker could read arbitrary files, create new ones, and conduct DoS attacks. The following screenshot provides an example of a vulnerability detected by the analyzer and a test exploit that proves the vulnerability's exploitability.

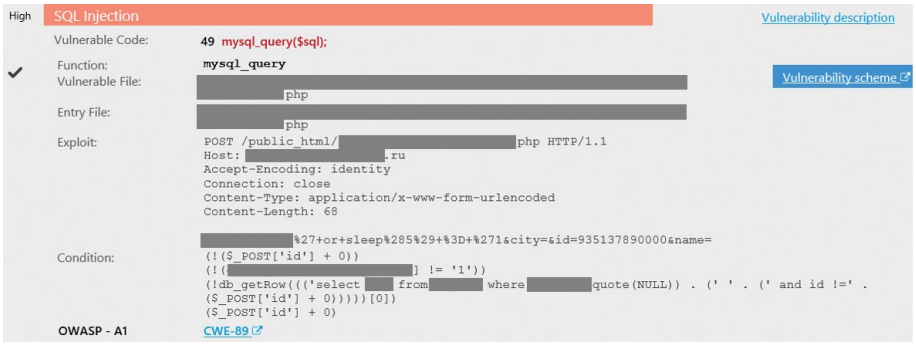

Figure 42. Example of SQL Injection detection

High-severity XML External Entity vulnerabilities were less common this year but not unheard of. They allow attackers to read arbitrary files or target corporate LAN resources. An example of this vulnerability is shown in the following screenshot.

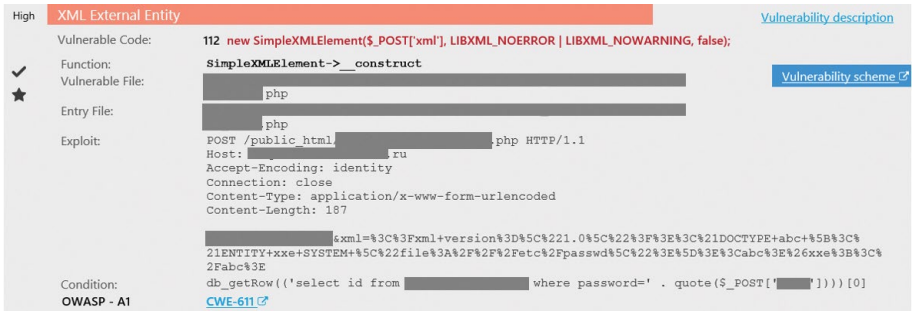

Figure 43. Example of XXE detection

Automated analysis revealed many other flaws in the source code of the examined applications: a hard-coded password, one-way unsalted hash function, and static random number generator were among the findings.

Overall, the results confirm that web application security assessment must be implemented throughout the entire software lifecycle. Automated source code testing allows identifying the maximum number of coding errors in the shortest possible time, including critical mistakes causing high-severity threats, which if left unfixed can provide a tempting target for attackers.

CONCLUSIONS

Although the percentage of web applications with high-severity vulnerabilities improved year-over-year, overall security remains weak. More than 50 percent of web applications have high-severity vulnerabilities, and this value rises sharply if an attacker has access to source code. By exploiting the vulnerabilities we detected, attackers could obtain large amounts of sensitive information including application source code and users' personal data, even from the websites of banks and government institutions. Users cannot rest safe: almost all applications can be exploited by attackers to target users.

We also found that web application vulnerabilities are the easiest way to penetrate the corporate LAN. About a quarter of the tested websites could be used by attackers to attack internal company systems.

Source-code analysis is much more effective than methods without access to the application code. Moreover, performing such code analysis during development significantly improves the security of the final application. Automated tools for source-code analysis should be used at multiple stages of development, because analyzers are much quicker than manual analysis.

We found that production web applications were more vulnerable than test applications. This underscores the need to perform security analysis not only during development, but when an application is already in production. Preventive protection measures, in the form of a web application firewall (WAF), are essential for keeping production systems safe.

Despite the modest improvements seen in 2016, web application security clearly still requires more attention. The results demonstrate the need to implement application security processes throughout the entire software lifecycle, both on the part of developers and system administrators responsible for secure operation. Only comprehensive security measures—including secure development procedures, preventive protection, and regular web application security testing—can minimize risks and provide a strong level of security.

About Positive Technologies

Positive Technologies is a leading global provider of enterprise security solutions for vulnerability and compliance management, incident and threat analysis, and application protection. Commitment to clients and research has earned Positive Technologies a reputation as one of the foremost authorities on Industrial Control System, Banking, Telecom, Web Application, and ERP security, supported by recognition from the analyst community. Learn more about Positive Technologies at ptsecurity.com.

POSITIVE TECHNOLOGIES

info@ptsecurity.com ptsecurity.com