



# WEB APPLICATION VULNERABILITIES

## STATISTICS FOR 2017

## CONTENTS

- Introduction..... 3
- 1. Materials and methods..... 3
- 2. Executive summary..... 4
- 3. Client snapshot..... 4
- 4. Trends ..... 5
- 5. Manual web application security assessment ..... 6
  - 5.1. Most common vulnerabilities ..... 7
  - 5.2. Analysis of threats and security levels ..... 8
  - 5.3. Analysis of development tools ..... 10
  - 5.4. Vulnerabilities in test and production applications..... 11
  - 5.5. Comparison of test techniques: black box, gray box, and white box..... 12
- Conclusion..... 14

## INTRODUCTION

Web applications are the rule, not the exception, in the modern economy. Governments provide services to citizens, media outlets publish news, IT and telecom companies advertise and sell their products and services, and businesses of all stripes manage internal processes using web applications.

But at the same time, the issue of security is often a mere afterthought. We regularly read about the latest hack of some company's web application: critical data is stolen, users are attacked, management apologizes, and the company is left to count its losses. Could all that have been avoided? We believe so.

Every year, Positive Technologies specialists assess the security of web applications and online banking systems, perform penetration testing of corporate infrastructures and wireless networks, carry out automated scanning, and evaluate staff security awareness at companies all over the world. This report is based on the results of web application testing performed in 2017, which are compared to equivalent data from 2016.

## 1. MATERIALS AND METHODS

This report includes data from comprehensive security assessments of 23 web applications tested in 2017. Results of penetration testing, automated scanning, and testing of e-banking systems are not included here; this information is reviewed in the relevant reports.

The security level of each application was measured using automated tools and manually using a combination of black-, gray-, and white-box methods. 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. White-box scanning refers to testing that makes use of all relevant information about the application, including its source code.

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, since these threats are implemented as part of other attacks.

The statistics in this report include only code and configuration vulnerabilities. Other widespread security weaknesses, such as failure to install software updates, are not considered here.

The severity of vulnerabilities was graded in accordance with the Common Vulnerability Scoring System ([CVSS v.3](#)). Based on the CVSS score, our experts assigned each vulnerability a severity level: high, medium, or low.

## 2. EXECUTIVE SUMMARY

### Almost half of web applications are vulnerable to unauthorized access

In 2017, our experts demonstrated that 48 percent of tested web applications were not protected from unauthorized access. In addition, the opportunity to gain full control was available in 17 percent of tested applications.

### All analyzed web applications contained vulnerabilities

Our experts found vulnerabilities in all tested web applications: 52 percent of applications contained high-severity vulnerabilities—and every application had medium-severity vulnerabilities.

### Code flaws cause 65 percent of vulnerabilities

The ten most common vulnerabilities in 2017 included four critical ones: SQL Injection, OS Commanding, Path Traversal, and XML External Entities. All these vulnerabilities are caused by source code defects.

### Personal data leakage in 44 percent of applications

An attacker can obtain personal data from 44 percent of applications processing such data. Finance, e-commerce, and telecom companies, among others, were affected. The threat of leakage of critical information was present in 70 percent of applications.

### High-severity vulnerabilities were found in 60 percent of PHP applications

Across all languages and frameworks, testers detected an average of two critical vulnerabilities per web application. And regardless of the development tool, almost every application contained one or more of the most common vulnerabilities.

### Testbeds contain more vulnerabilities than production systems

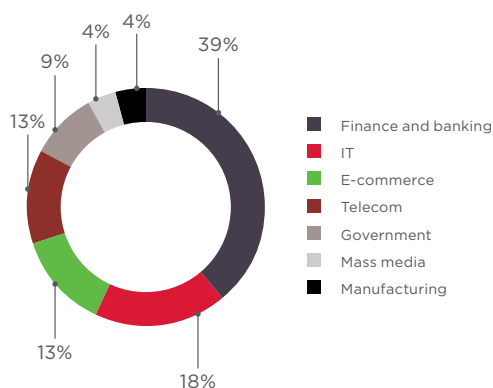
As we found in 2017, testbeds are more vulnerable. On average, testers detected five times more high-severity vulnerabilities in testbeds than in production systems.

### Access to source code makes assessment more effective

Manually analyzing source code, our experts detected high-severity vulnerabilities in 100 percent of applications. Black-box testing without such access revealed such vulnerabilities in only 35 percent of web applications.

## 3. CLIENT SNAPSHOT

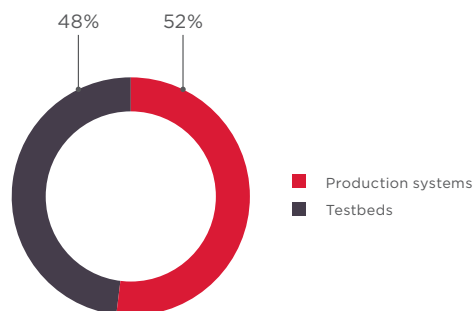
In 2017, Positive Technologies assessed the security of web applications from a variety of sectors: finance, e-commerce, manufacturing, telecom, IT, and government.



Industries to which tested web applications belonged

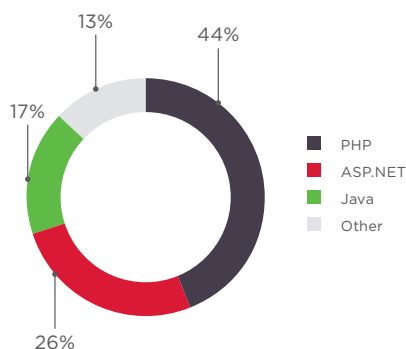
The following statistics are given without per-industry breakdowns.

Security assessment was performed for both production (52%) and testbed (48%) systems in either the final development or release stage.



Relative percentage of testbed and production systems

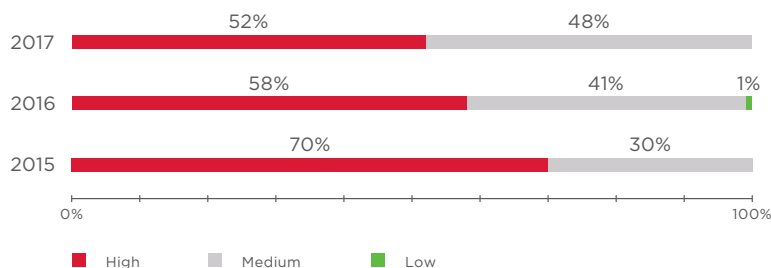
A plurality (44%) of tested web applications were written in PHP. Java and ASP.NET accounted for 17 and 26 percent, respectively. Other development tools, such as Python, Node.js, and Ruby on Rails, increased from 7 to 13 percent.



Development tools (percentage of web applications)

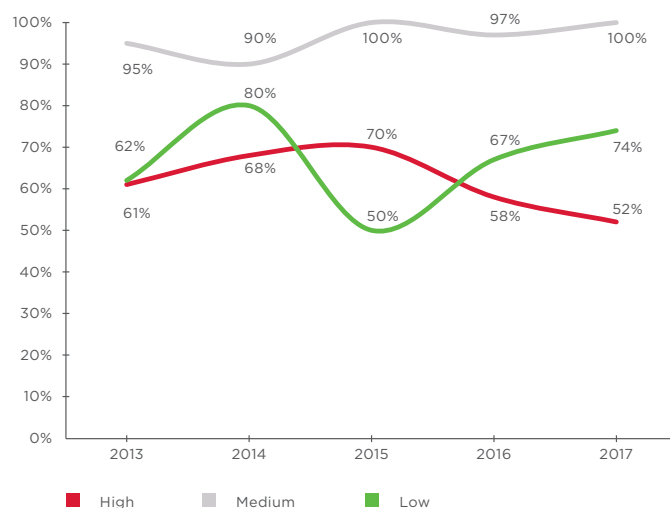
## 4. TRENDS

Every tested web application contained vulnerabilities of various severity levels. However, we observe an encouraging trend: the percentage of web applications containing critical vulnerabilities has decreased for the second year in a row. In 2017, high-severity vulnerabilities were detected in 52 percent of applications.



Websites by maximum severity of vulnerabilities found

Every tested application contained medium-severity vulnerabilities. This value has hovered between 90 and 100 percent for several years. As in the previous year, the number of applications containing low-severity vulnerabilities has increased: in 2017, these vulnerabilities were found in 74 percent of tested applications.



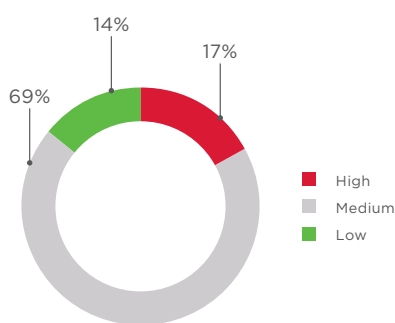
Websites by vulnerability severity

In 2017, security assessment revealed that 21 percent of web applications used outdated software containing vulnerabilities of low, medium, and high severity. These vulnerabilities are not included in our statistics, since they are not caused by the web application itself.

However, if exploited, such vulnerabilities can pose a critical threat to a web application. The most common detected vulnerabilities were older versions of web servers or content management systems. For example, security assessment of one website revealed that any external attacker could use a publicly accessible exploit to perform a Denial of Service<sup>1</sup> attack, because the website used an outdated version of the Apache web server that contained vulnerability [CVE-2011-3192](#) disclosed in 2011. Another example: testers found a vulnerable version of ImageMagick, which was intended to convert images uploaded by users on their account pages. Experts used a publicly accessible tool<sup>2</sup> to exploit vulnerability [CVE-2017-15277](#), which allows obtaining a part of operating system memory on a remote host. As a result, information about the application's directories and files was gained.

## 5. MANUAL WEB APPLICATION SECURITY ASSESSMENT

In 2016, we had observed a significant decrease in the proportion of high-severity vulnerabilities. Unfortunately, 2017 could not keep up this positive trend and these vulnerabilities rebounded to 17 percent. Most detected vulnerabilities (69%) were again of medium severity, and 14 percent were classified as low-severity vulnerabilities.

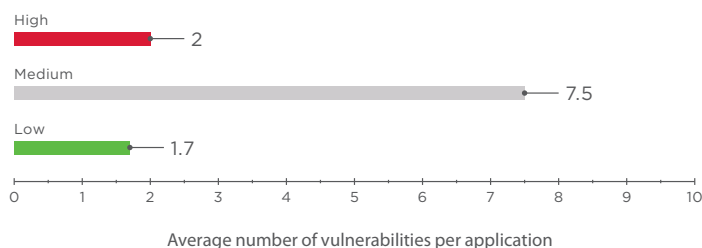


Vulnerabilities by severity

<sup>1</sup> [exploit-db.com/exploits/17696/](https://exploit-db.com/exploits/17696/)

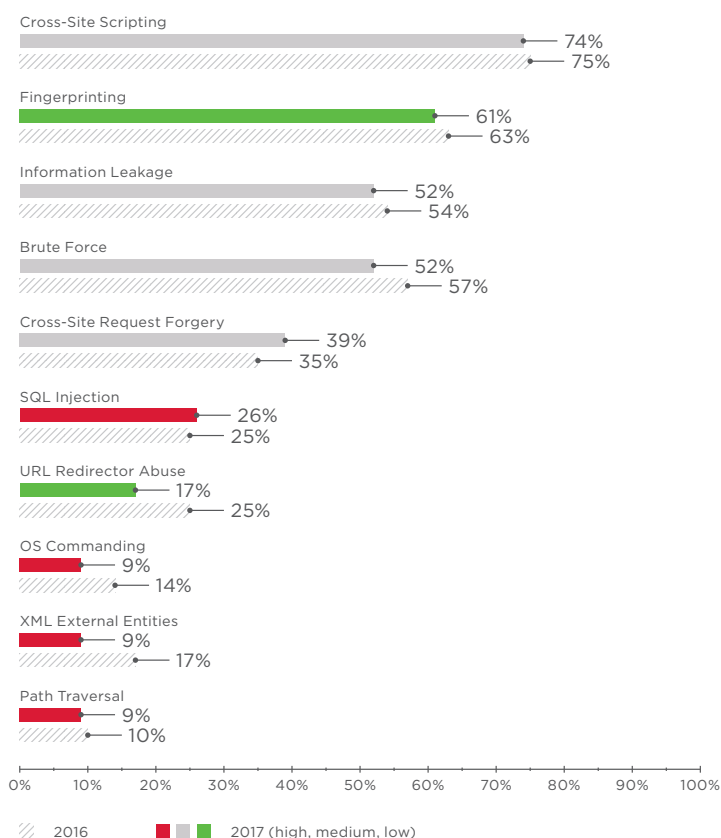
<sup>2</sup> [github.com/neex/gifoeb](https://github.com/neex/gifoeb)

The average number of critical vulnerabilities per web application remained almost the same: 2 compared with 2.1 in 2016. Although all web applications tested in 2017 contained medium-severity vulnerabilities, the average number of these vulnerabilities fell from 17.3 to 7.5 per application. The number of low-severity vulnerabilities remained practically unchanged (1.7 compared to 1.8 in 2016).



## 5.1. Most common vulnerabilities

The following chart displays the vulnerabilities most commonly detected by our experts during manual security assessment of web applications. The most common vulnerabilities are the same as in 2016.



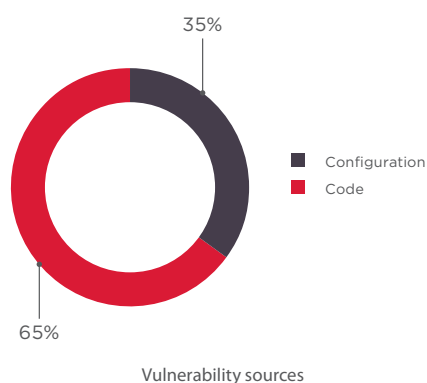
Most common vulnerabilities detected in manual testing (percentage of tested web applications)

Cross-Site Scripting, as usual, occupies the top slot: this medium-severity vulnerability was found in 74 percent of applications. Other common vulnerabilities that also allow attacks against web application users are Cross-Site Request Forgery and URL Redirector Abuse. Four of the top ten vulnerabilities are of high severity. Experts demonstrated that one out of every four tested web applications is vulnerable to SQL Injection: an attacker is able to obtain sensitive information including user credentials from the database management system. Such vulnerabilities as OS Commanding, XML External Entities, and Path Traversal were in 9 percent of web applications.



Server-side and client-side vulnerabilities were present in equal proportion. Server-side vulnerabilities included Information Leakage and Insufficient Transport Layer Protection. Client-side vulnerabilities included Cross-Site Scripting, Cross-Site Request Forgery.

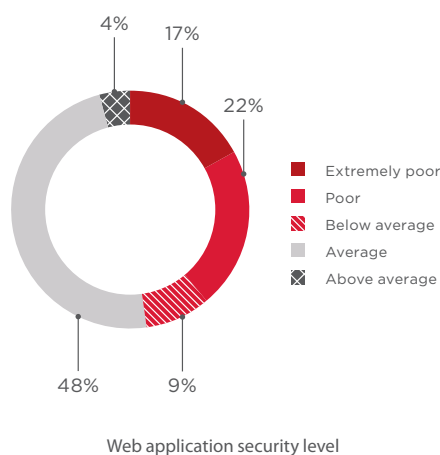
The majority of flaws (65%) are in software code, as the result of errors in application development. Incorrect web server configuration accounted for the remaining third of security flaws.



Most of the detected web application security flaws can be avoided by implementing a secure software development lifecycle, including security assessment during code development.

## 5.2. Analysis of threats and security levels

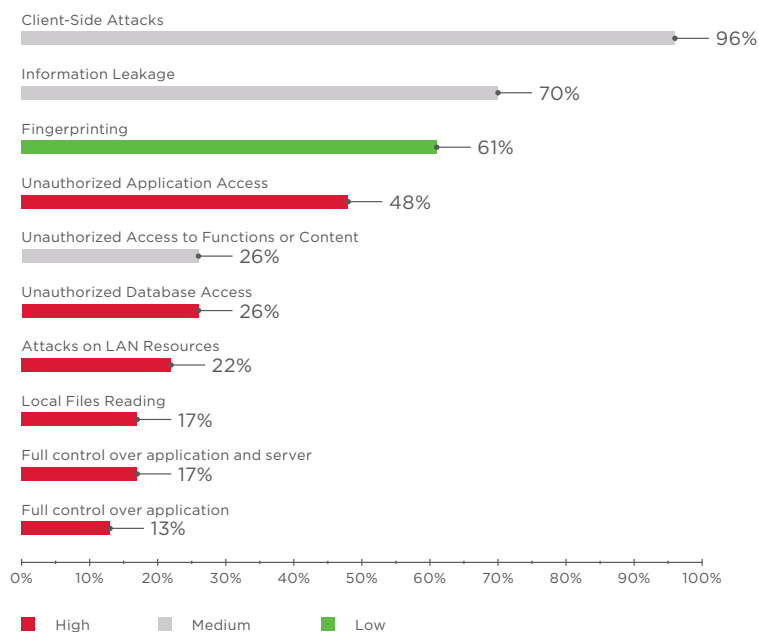
Web application assessment includes evaluation of the security level on a scale from "extremely poor" to "acceptable." "Extremely poor" means presence of numerous critical vulnerabilities that, for example, allow execution of OS server commands by any external attacker or disclosure of sensitive information. Depending on the number of critical vulnerabilities and complexity of vulnerability exploitation, the security level may vary from "extremely poor" to "below average." In 2017, almost half of tested applications (48%) were evaluated as "average." Almost half of web applications (48%) were within the range from "below average" to "extremely poor"; in 2016, the corresponding figure was 56 percent. The security level of 17 percent of applications was evaluated as "extremely poor," which is almost the same as in 2016 (16%). None of the tested web applications was assessed as having "acceptable" protection. The overall security level of web applications still leaves much to be desired.



The frequency of threats was in keeping with 2016. Attacks against users can be performed in 96 percent of web applications. Potentially critical leaks of data, including personal data, increased. Unauthorized access to an application can be obtained almost in half of test cases (48%). One in every four web applications can be a vector for intranet penetration.

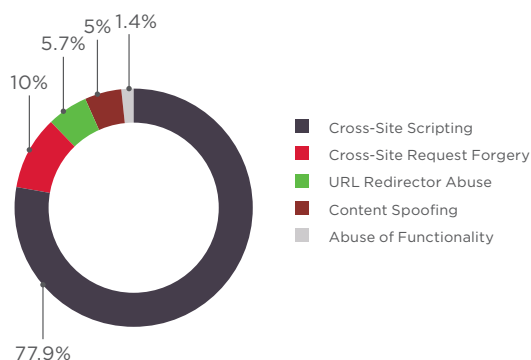


Such threats as "Full control over application and server" and "Full control over application" deserve special attention. To obtain full control over the application and server, it is necessary to find a vulnerability that, if exploited, would allow executing commands on the attacked application server, such as OS Commanding or Arbitrary File Upload. These flaws were found in 17 percent of web applications. "Full control over application" means obtaining maximum privileges in the application without gaining control over the company server, for example, thanks to an accessible copy of a database that contains the cleartext credentials for a web application administrator account. Such flaws were found in 13 percent of web applications.



Most common threats (percentage of tested web applications)

Looking separately at vulnerabilities that allow attacks against web application users, the majority are Cross-Site Scripting vulnerabilities (77.9%). Other vulnerabilities include Cross-Site Request Forgery (10%) and URL Redirector Abuse (5.7%). As already mentioned, these vulnerabilities were among the 10 most common vulnerabilities in 2017.



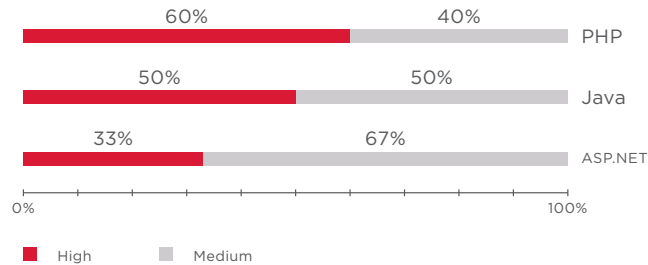
Relative frequency of vulnerabilities allowing attacks against users

Experts demonstrated access to web application source code in 4 percent of web applications. Obtaining source code enables an attacker to discover other vulnerabilities in the application and gather information for planning and conducting additional attacks.

The percentage of web applications with accessible personal data grew significantly compared to 2016. Experts demonstrated that personal information could be obtained in 44 percent of applications that process user data. This data is highly valuable for attackers and can be used for planning attacks against users.

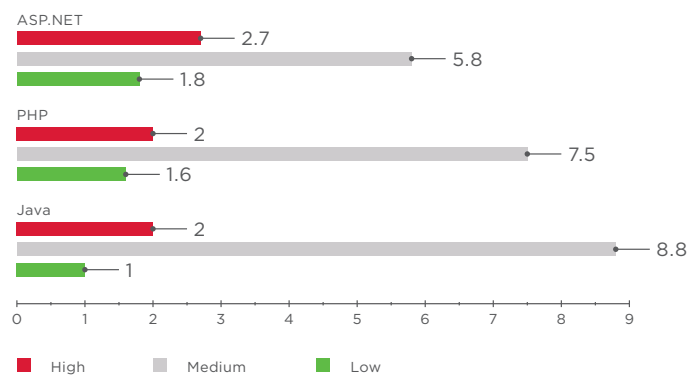
### 5.3. Analysis of development tools

Among web applications developed using PHP, 60 percent contained critical vulnerabilities. Java- and ASP.NET-based applications were slightly less affected, at 50 and 33 percent respectively. Thus, we can observe a positive trend for three consecutive years: the percentage of Java- and ASP.NET-based applications with critical vulnerabilities has been decreasing.



Web applications by maximum vulnerability severity

The average number of high-severity vulnerabilities for ASP.NET increased from 2.1 to 2.7. PHP and Java web applications had an average of two high-severity vulnerabilities (as compared to 2.8 and 2.1 in 2016, respectively). As in 2016, Java-based applications tended to contain slightly more medium-severity vulnerabilities. ASP.NET-based applications again contained the fewest such flaws.



Average number of vulnerabilities per application

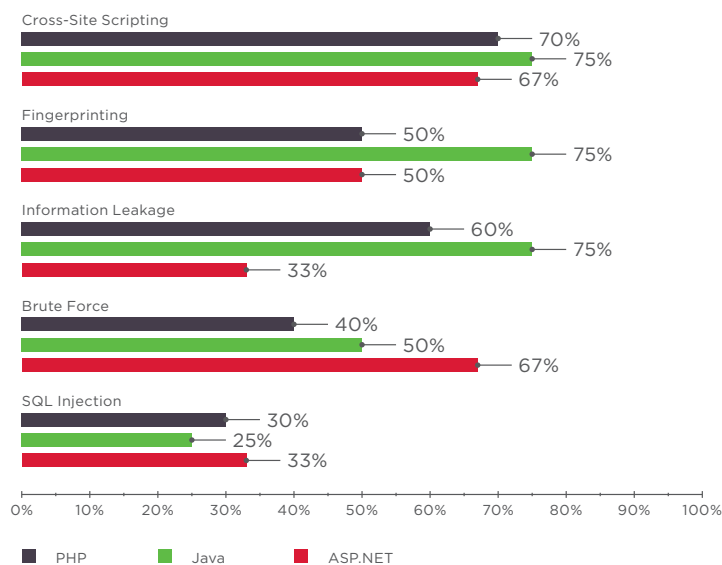
The following table displays the most common vulnerabilities in web applications created with a particular development tool.

Most common vulnerabilities, by development tool

PHP	% of websites	Java	% of websites	ASP.NET	% of websites
Cross-Site Scripting	70%	Cross-Site Scripting	75%	Cross-Site Scripting	67%
Information Leakage	60%	Fingerprinting	75%	Brute Force	67%
Fingerprinting	50%	Information Leakage	75%	Fingerprinting	50%
Brute Force	40%	Brute Force	50%	Cross-Site Request Forgery	50%
SQL Injection	30%	Insufficient Authorization	50%	SQL Injection	33%
Cross-Site Request Forgery	30%	Content Spoofing	50%	URL Redirector Abuse	33%
Application Misconfiguration	20%	XML External Entities	25%	Insufficient Authorization	17%
OS Commanding	10%	SQL Injection	25%	XML External Entities	17%
URL Redirector Abuse	10%	Cross-Site Request Forgery	25%	OS Commanding	17%
Path Traversal	10%	Deserialization of Untrusted Data	25%	Insecure indexing	17%

Cross-Site Scripting is still the most common vulnerability for all languages, varying from 67 to 75 percent. Information Leakage, Fingerprinting, and Brute Force are also common across the board.

The top 10 critical vulnerabilities include SQL Injection (for all programming languages), XML External Entities (Java, ASP.NET), OS Commanding (PHP, ASP.NET), as well as Path Traversal (PHP), Deserialization of Untrusted Data (Java), and Insufficient Authorization (ASP.NET).

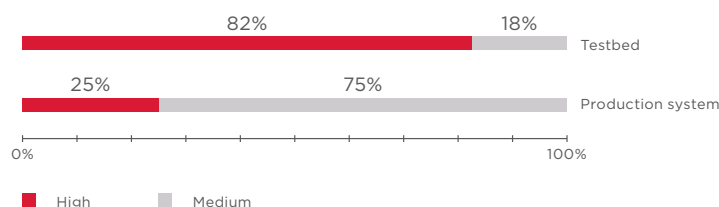


Percentage of web applications containing the most common vulnerabilities

Almost all tested web applications, regardless of the development tool, contained some of the most common vulnerabilities.

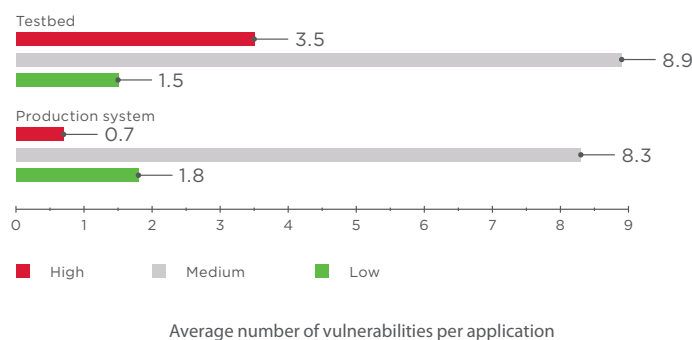
## 5.4. Vulnerabilities in test and production applications

Security assessment was performed for both test and production web applications. Systems that are not yet in production are more vulnerable to high-severity vulnerabilities: 82 percent of such applications contained critical security flaws. For production systems, this figure has declining for three consecutive years and now stands at 25 percent, proving that developers are paying more attention to secure development and fixing vulnerabilities before putting an application online. Although the current situation is far from satisfactory, we hope that this progress will continue.



Applications compared by maximum vulnerability severity

Comparing the number of detected vulnerabilities, we see that production systems are better protected. The average number of high-severity vulnerabilities in test applications is 3.5, almost five times more than in production applications (0.7). An important caveat: most testbeds were assessed using white-box testing, which is able to detect more security flaws than other methods.

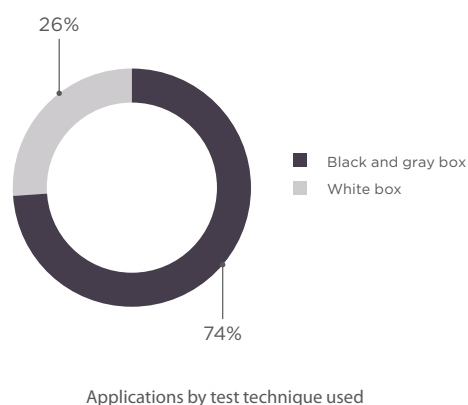


Both test and production systems contain many critical vulnerabilities. Therefore, it is important that a secure software development lifecycle should be implemented from the very start of the development process. Hurriedly fixing flaws right before putting an application into operation is a poor replacement for a systematic approach.

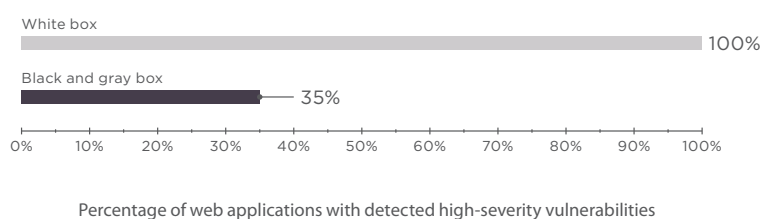
### 5.5. Comparison of test techniques: black box, gray box, and white box

In 2017, most assessments used black- or gray-box testing (74%); source code was provided for white-box testing in a quarter of cases.

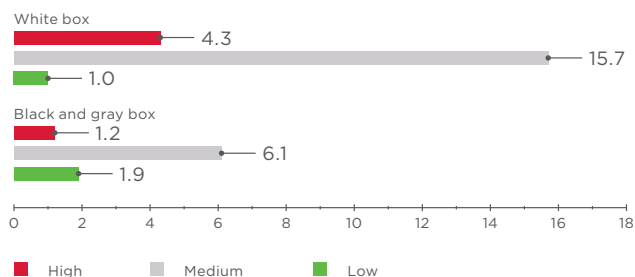
One web application tested by our experts contained dozens of critical vulnerabilities, including SQL Injection; another application harbored hundreds of medium-severity configuration flaws. We have omitted these test cases from our statistical averages because of their disproportionate effect.



For the third year in a row, comparative analysis of test techniques—black box, gray box, and white box—confirms that white-box testing is more effective. Although medium-severity vulnerabilities were detected in all web applications regardless of the test technique, results for high-severity vulnerabilities were different: gray- and black-box testing found high-severity vulnerabilities in 35 percent of applications, whereas white-box testing did so in 100 percent of applications. So while attackers can often find a way in even without access to source code, the chances of discovering and exploiting critical vulnerabilities jump dramatically if this code becomes available.

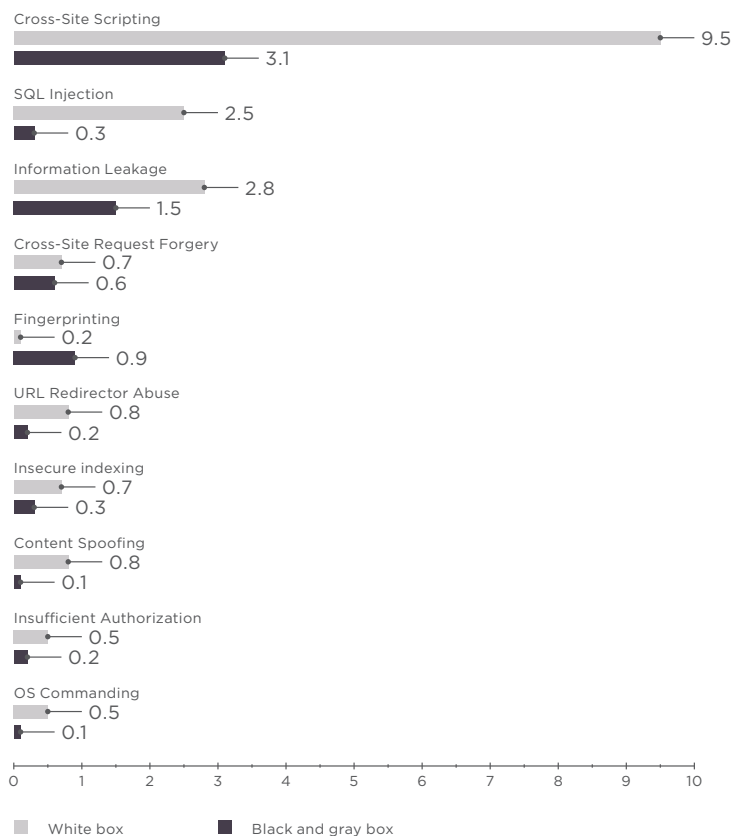


The average number of vulnerabilities per web application detected without access to source code was 1.2 in 2017, while white-box testing revealed 4.3 vulnerabilities per application. For medium-severity vulnerabilities, the rate of detection per application was 15.7 vulnerabilities with access to source code, and 6.1 without it.



Average number of detected vulnerabilities per web application

With access to source code, experts detected eight times more SQL Injection and five times more OS Commanding vulnerabilities on average. White-box testing was also more effective for uncovering such vulnerabilities as Cross-Site Scripting, Information Leakage, URL Redirector Abuse, and Insufficient Authorization.



Average number of detected vulnerabilities per web application

## CONCLUSION

To sum up our findings: the average level of web application security remains poor. Every application tested had flaws of various severity levels. High-severity vulnerabilities can be exploited in more than half of applications for access to sensitive data, execution of commands on a server, and total control of the system. Web applications in any sector—from e-commerce to government—can be successfully attacked. Only 4 percent of tested applications were free of threats to their users. In one out of four cases, attacks against web applications can yield personal data, which is highly valuable for attackers.

Another motivation for attackers is to infect users with malware. This method was used to spread the Bad Rabbit ransomware: attackers hacked web applications belonging to media outlets and masked malware as an Adobe Flash Player update installer. In another case, the Cobalt group<sup>3</sup> found vulnerabilities in the web applications of banks' business partners. The group then sent phishing emails from the addresses of these partners to employees at the targeted banks. This is why we say that every web application can be at risk: even if it is not the target, it can be used as a link in the attack chain. And if a website is critical for a company, attacks against clients may cause both reputational and financial losses.

The results for 2017 demonstrate the necessity of performing regular web application security assessments. White-box testing (with source code analysis) is preferable: it is more effective than other techniques. The significant expenses associated with application revisions and recoding can be avoided if code security is assessed during the development process ([Secure Software Development Lifecycle](#)).<sup>4</sup> A proactive approach works to prevent vulnerabilities at the earliest stages.

Another important preventive security measure is a web application firewall (WAF), which protects from known attacks at the level of applications and business logic. A WAF also detects attempts to exploit zero-day vulnerabilities, prevents attacks against users, and analyzes and correlates events in order to build attack chains. Performing these tasks requires advanced implementations of normalization, heuristics, machine learning, and behavioral analysis.

Security issues are still not given their due: every year we observe the same errors being made by developers and system administrators. By implementing security as a process throughout the web application lifecycle, perhaps businesses can start improving this situation in earnest.

---

<sup>3</sup> [ptsecurity.com/upload/corporate/ww-en/analytics/Cobalt-2017-eng.pdf](https://ptsecurity.com/upload/corporate/ww-en/analytics/Cobalt-2017-eng.pdf)

<sup>4</sup> [owasp.org/index.php/OWASP\\_Secure\\_Software\\_Development\\_Lifecycle\\_Project](https://owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project)

---

### 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](https://ptsecurity.com).

© 2018 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.