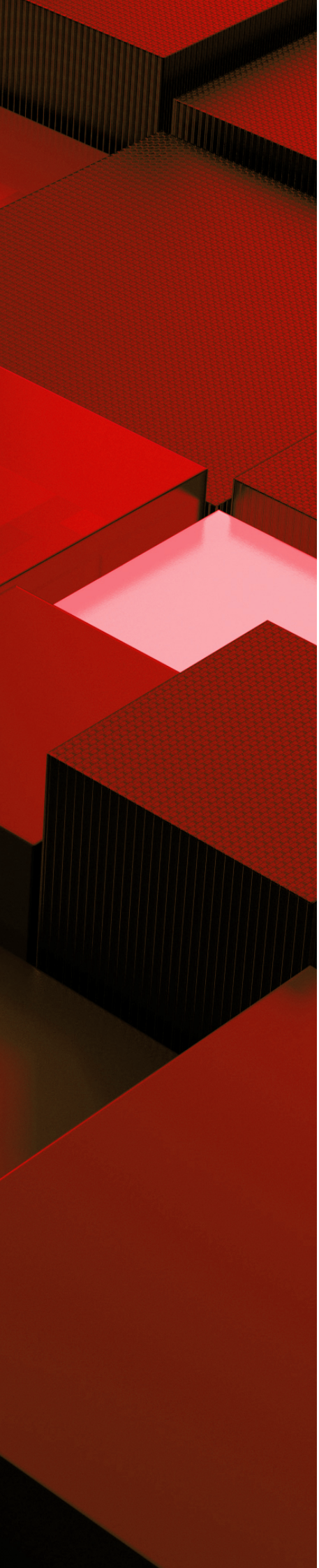
Web application vulnerabilities and threats: statistics for 2019

ptsecurity.com

Contents



Executive summary

The overall security of web applications has continued to improve, but still leaves much to be desired.

Key takeaways regarding web applications:

- **Hackers can attack users in 9 out of 10 web applications.** Attacks include redirecting users to a hacker-controlled resource, stealing credentials in phishing attacks, and infecting computers with malware.
- **Unauthorized access to applications is possible on 39 percent of sites.** In 2019, full control of the system could be obtained on 16 percent of web applications. On 8 percent of systems, full control of the web application server allowed attacking the local network.
- **Breaches of sensitive data were a threat in 68 percent of web applications.** Most breachable data was of a personal nature (47% of breaches) or credentials (31%).

Vulnerability statistics:

- **82 percent of vulnerabilities were located in application code.**
- **The average number of vulnerabilities per web application fell by a third compared to 2018.** On average, each system contained 22 vulnerabilities, of which 4 were of high severity.
- **One out of five vulnerabilities has high severity.**

Trends

The percentage of web applications containing high-risk vulnerabilities in 2019 fell significantly, by 17 percentage points compared to the prior year. The average number of severe vulnerabilities per web application also fell, by almost one third.

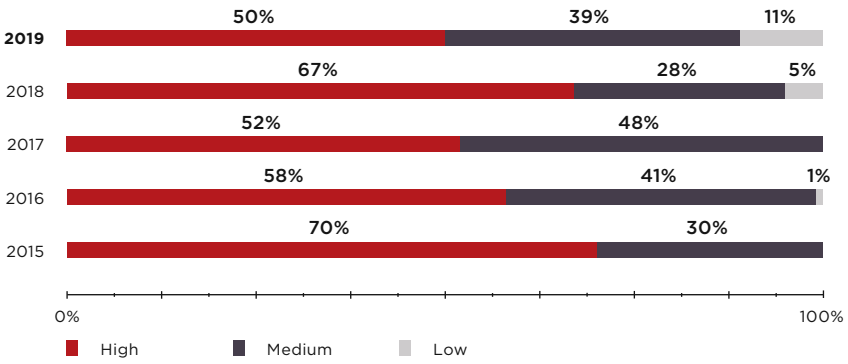


Figure 1. Websites by maximum severity of vulnerabilities found

The last five years show a reduction in the percentage of sites containing severe vulnerabilities. This is an encouraging sign consistent with an overall improvement in security.

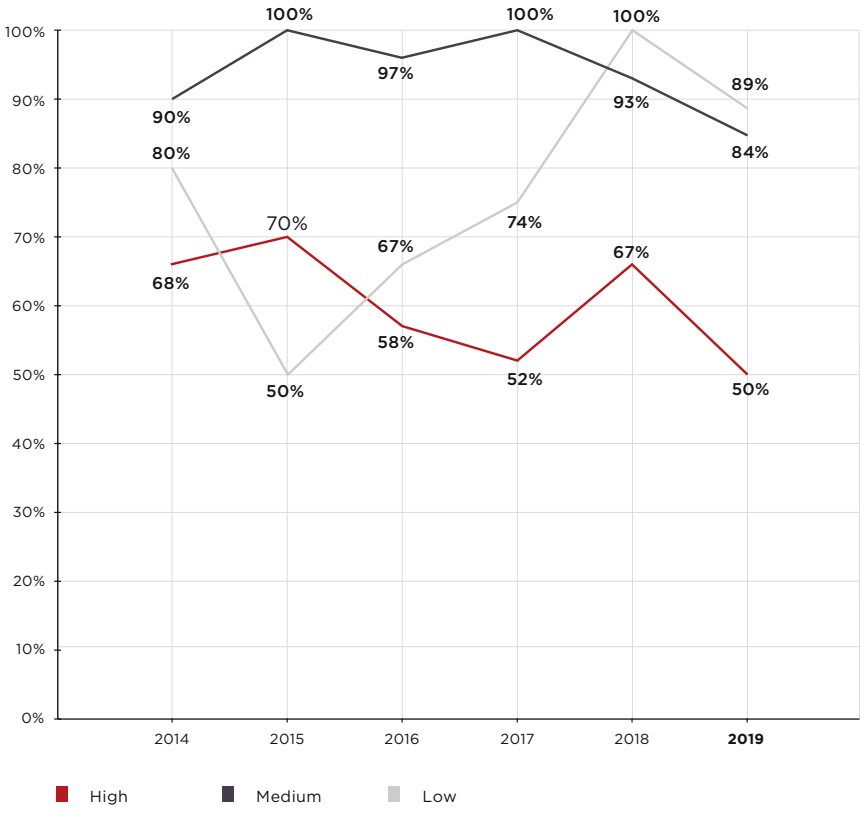


Figure 2. Websites by vulnerability severity

Assessment of web application security

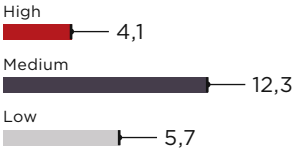


Figure 3. Average number of vulnerabilities per application

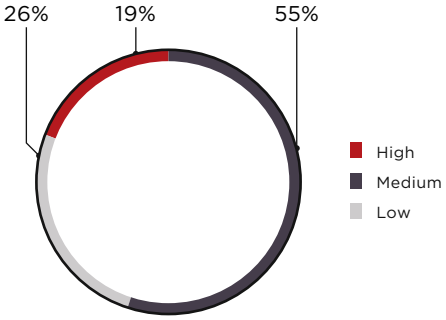


Figure 4. Vulnerabilities by severity

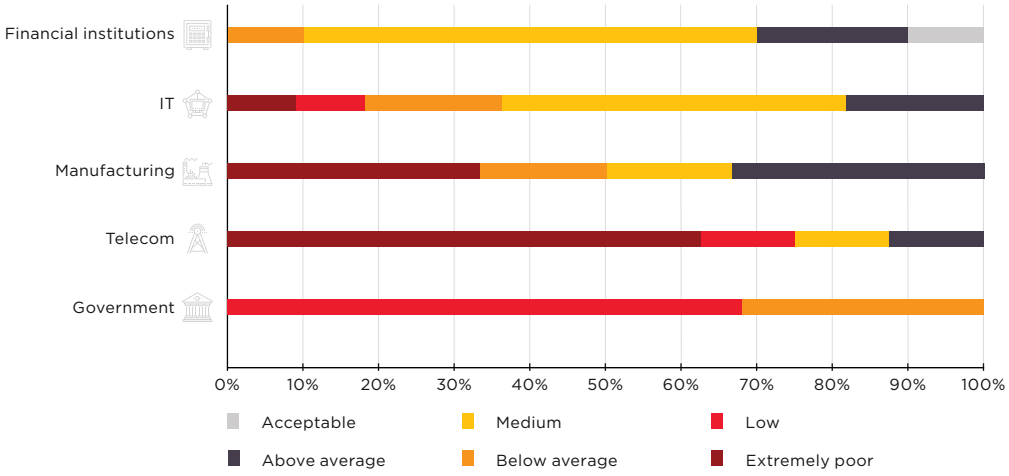


Figure 5. Vulnerabilities of various severity levels, by industry

Web application security is measured by our experts in the course of testing and assessment. The level they assign depends on the potential impact on the particular system in question, in the context of the kinds of information processed on that system.

Most common vulnerabilities

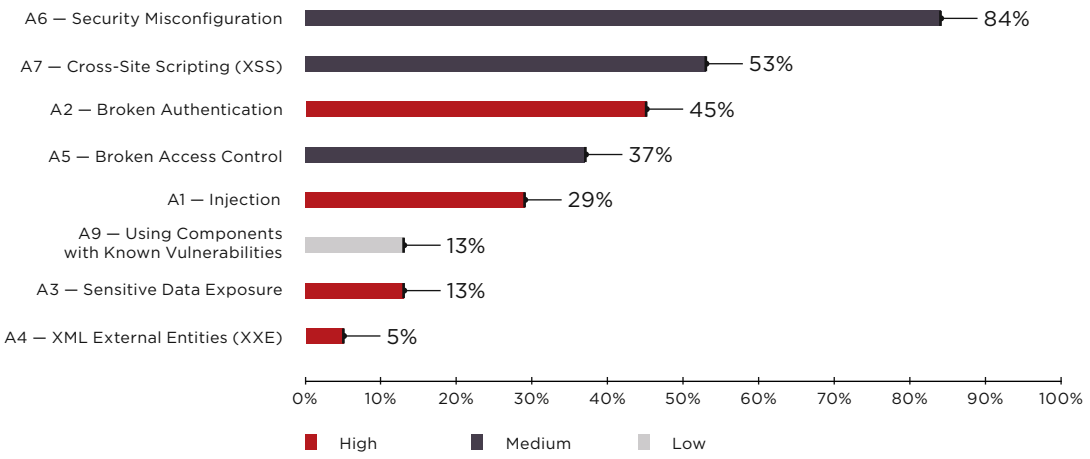


Figure 6. Most common OWASP Top 10 vulnerabilities (percentage of web applications)

The most commonly encountered web application vulnerabilities in 2019 involved Security Misconfiguration. One out of every five tested applications contained vulnerabilities allowing the hackers to attack a user session, such as sensitive cookies without the HttpOnly and Secure flags. Attackers can use such flaws to perform Cross-Site Scripting (XSS) in order to capture the user's session identifier and impersonate the user in the application.

Broken Authentication was found in 45 percent of web applications. Almost a third of such vulnerabilities consist of failure to properly restrict the number of authentication attempts. An attacker can exploit this to bruteforce credentials and access the web application. For instance, one of the applications could be accessed with administrator rights after only 100 attempts.

Password-only authentication is a contributing factor in most authentication attacks. Password age and complexity requirements that previously were the "gold standard" now are found to undermine security. **According to the latest recommendations by the NIST, organizations should move to multifactor authentication if they have not already.**

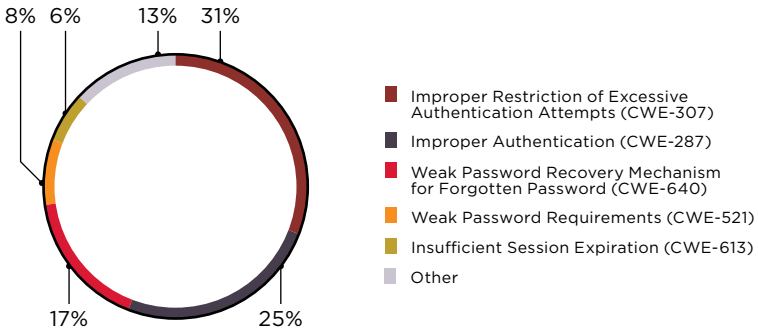


Figure 7. Vulnerabilities related to Broken Authentication

Every third application in 2019 had Broken Access Control. Bypassing access restrictions usually leads to unsanctioned disclosure, modification, or destruction of data. In one tested application, insecure authorization allowed changing the profile of any user. Positive Technologies specialists found out the username of the application administrator, replaced the corresponding email address in the user profile with an address of their own, and then used the standard password reset procedure to access the site with administrator rights.

It is usually possible to minimize authentication and authorization vulnerabilities by following the Secure Software Development Lifecycle (SSDLC) during web application development.

In addition to the Top 10 2017 vulnerabilities, OWASP points out a number of flaws to check for.[1] We found a third of web applications to be vulnerable to clickjacking (User Interface Misrepresentation of Critical Information, CWE-451). Another third were vulnerable to Cross-Site Request Forgery (CSRF). In a CSRF attack, the hacker uses specially crafted scripts to perform actions posing as a user logged in to a vulnerable web application. Imagine you are logged in to a website vulnerable to CSRF—an online bank, for example. You receive a phishing message with a link and click that link. Next, the hacker

In a clickjacking attack, the user is usually already on the attacker's site and tempted to click a button promising great discounts or the secret to eternal youth. On top of that button, the malefactor places a transparent HTML frame (iframe) belonging to a vulnerable site. So when the user clicks the button, this results in an unintended action on the vulnerable site, such as liking someone's photo. This quickly drives up likes, votes, and so on. One way to prevent this is to use the X-Frame-Options HTTP header.

1. owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

sends a specially crafted request to the vulnerable site (your online bank) to perform certain actions that the hacker wants (for instance, transfer funds to the hacker's accounts). The online bank will not be able to tell this unauthorized request from a legitimate one unless it uses protection from CSRF attacks. Protection usually involves requiring unique one-time keys (CSRF tokens), confirming authenticity (with a password, for example), making sure that the request is being made by an actual user (perhaps with CAPTCHA), or setting an additional SameSite flag for cookies.

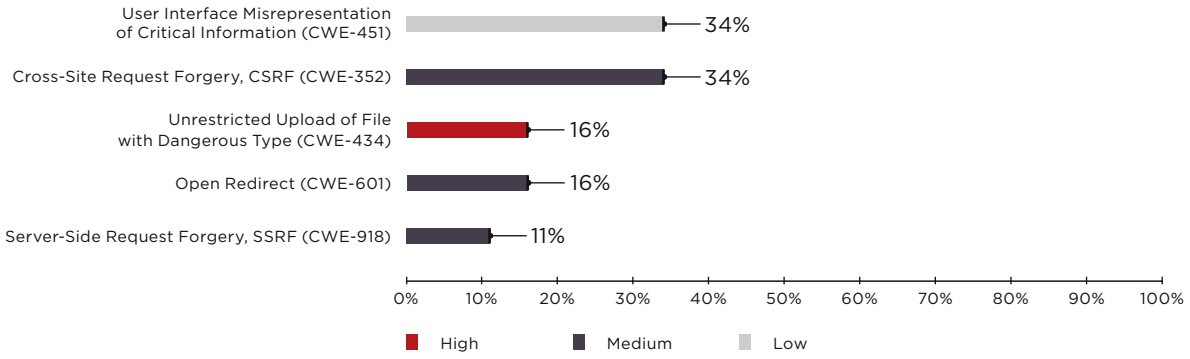


Figure 8. Common vulnerabilities not in the OWASP Top 10 (percentage of applications)

Threat analysis

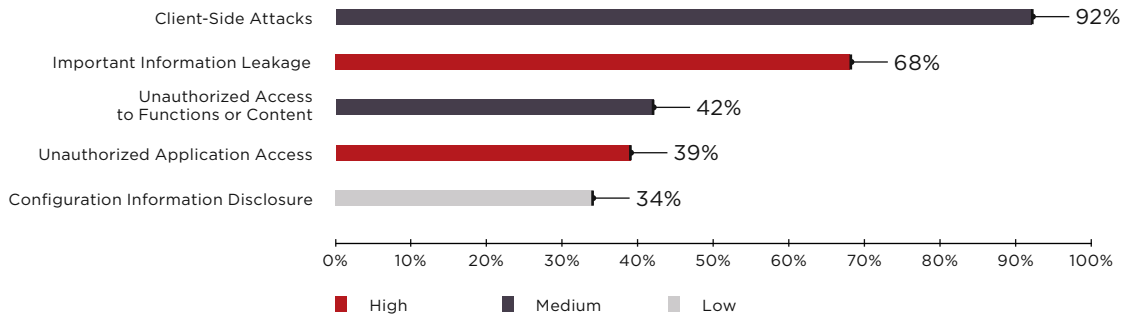


Figure 9. Top 5 most common treats (percentage of applications)

In spring 2019, owners of WordPress sites fell victim to a mass XSS attack targeting the Yuzo Related Posts plugin.

Attacks on clients remained a threat for nine out of every ten applications in 2019, just like in 2018. Cross-Site Scripting (XSS) remains one important cause. Attackers can infect computers with malware, stage phishing attacks to grab credentials, say, and perform actions posing as the user. As a general recommendation, web applications should sanitize all user input that is subsequently displayed in a browser, including HTTP request header fields such as User-Agent and Referer. Potentially unsafe characters that can be used in HTML page formatting must be replaced with their non-formatting equivalents. We also recommend using modern web application firewalls, since they are able to block cross-site scripting.

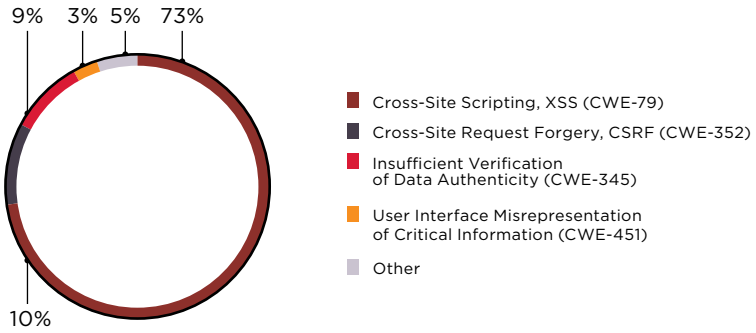


Figure 10. Vulnerabilities allowing attacks against clients

When important information is obtained by intruders, it's usually due to authorization and authentication failures in an application.

Breaches of important information are the second-most pressing threat to site security. In almost half of all breaches (47%), personal data was at risk. User credentials figured prominently as well (31%). As we can see from our analysis of 2019 cybersecurity incidents,[2] information is the prime target of hackers when they target organizations.

2. ptsecurity.com/ww-en/analytics/cybersecurity-threatscape-2019-q3/

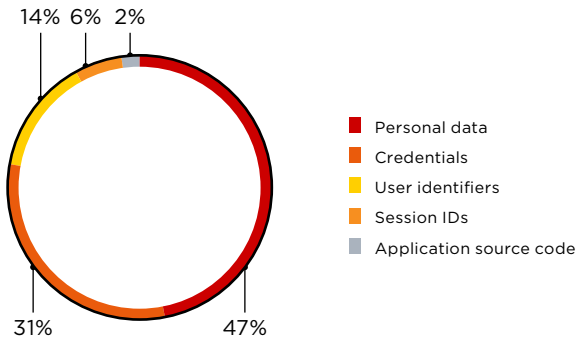


Figure 11. Breaches of sensitive data

Most dangerous threats

In 16%

of web applications, it is possible to gain full control

Attacks on LAN resources are possible

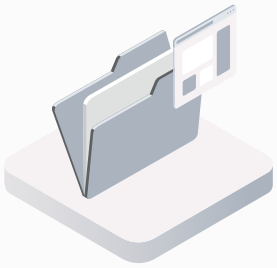
In 8%

of web applications

In 16 percent of web applications, severe vulnerabilities allowed taking control of both the application and the server OS.

For instance, access to a web application can be used to inject a JavaScript sniffer into its code and attack site users. Sniffers can steal both credentials and personal data, as well as payment card information. Attacks with JavaScript sniffers were the most dangerous attacks on individuals in 2018–2019. Since sniffers are injected into code, it takes white-box security analysis to discover them.

In a targeted attack against a company, web application vulnerabilities can help with gathering data about the company's internal network, such as the structure of the network segments, ports, and services. In many cases, hackers can even access internal network resources and the confidential data stored there.



Hackers can combine sets of stolen data and use them to attack other web resources, in so-called credential stuffing. In May 2019, such attacks affected half a million clients of two online stores.

Vulnerabilities in testbed and production applications

The percentage of production systems with high-risk vulnerabilities declined: 45 percent in 2019 compared to 71 percent in 2018. But this is still higher than in 2017, when the equivalent figure was 25 percent. Among testbed systems, the situation is the same as last year. The percentage of apps with high-risk vulnerabilities was 56 percent.

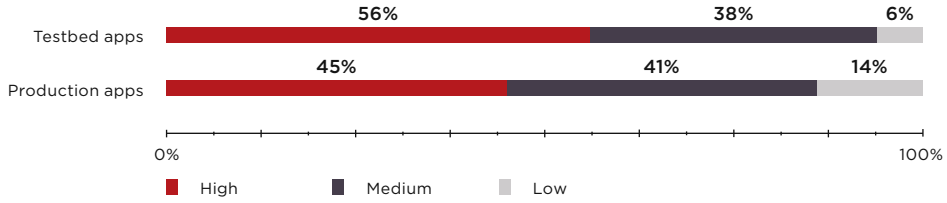


Figure 12. Most severe vulnerability found (percentage of applications tested)

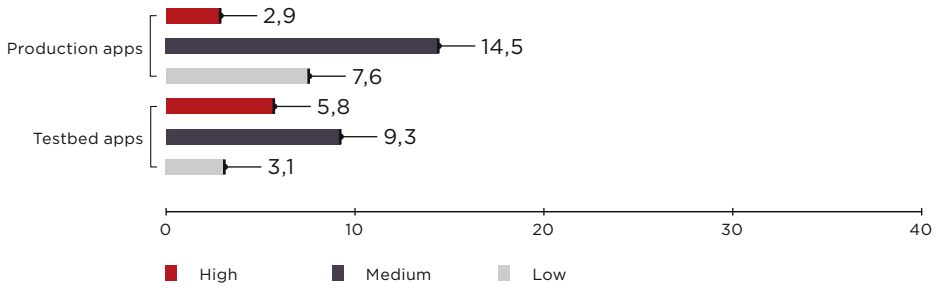
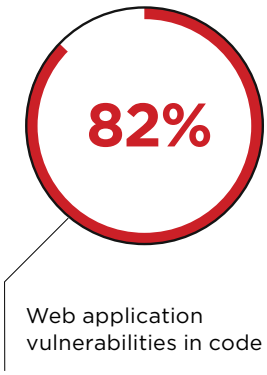


Figure 13. Average number of vulnerabilities per application

In 2019, production systems made up exactly 50 percent of all tested apps (versus 79 percent in 2018). During security assessment of production systems, companies are loath to risk disrupting their web applications. Therefore, not all tests are performed and it becomes impossible to demonstrate exploitation of some potential vulnerabilities. In cases such as these, demonstrating vulnerabilities on a testbed system is a good option.

We have repeatedly pointed out the importance of a Secure Software Development Lifecycle. Paying attention to security throughout development is always going to be more effective than a haphazard, after-the-fact approach. When developers race to patch vulnerabilities in a web application that is already live, shortcuts and mistakes inevitably result.



Web application vulnerabilities in code

White-box security assessment

In our experience, most site vulnerabilities are caused by errors in web application code. This is the main reason for providing testers with the source code for analysis, or else doing such analysis independently with a code analyzer as part of a Secure Software Development Lifecycle.

White-box security assessment is performed simultaneously by several specialists for maximum coverage and detection of as many vulnerabilities as possible. This work also includes manual and automated code analysis. Automated detection helps to speed up testing, but requires manual verification to rule out false positives. While manual methods take longer, the vulnerabilities detected will be real.

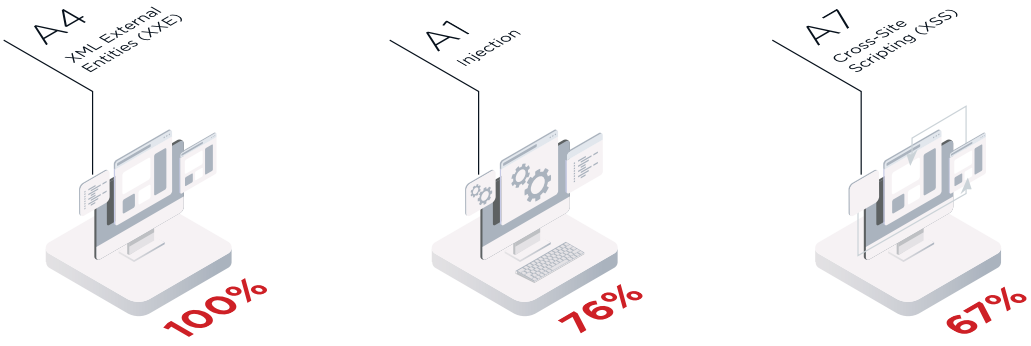
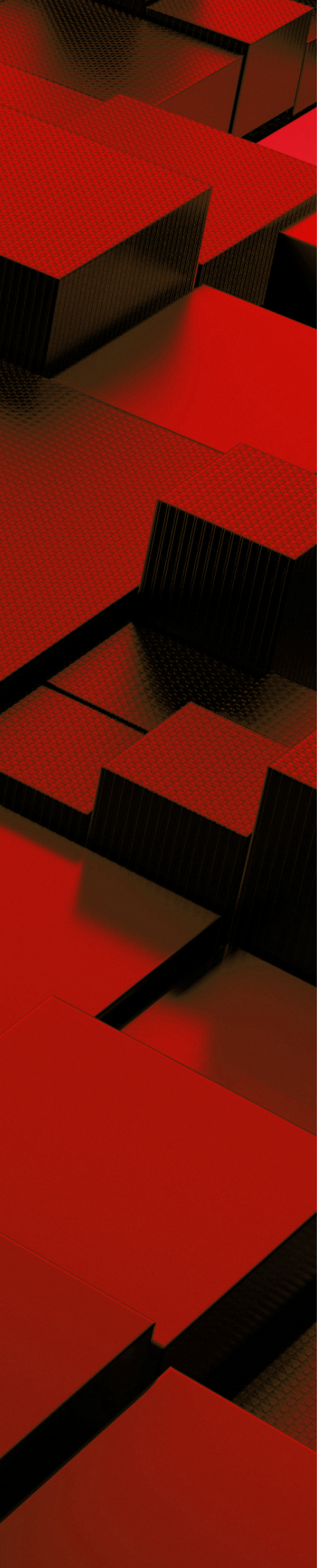


Figure 14. Percentage of OWASP Top 10 vulnerabilities detected by white-box testing

During security assessment of one web app, Positive Technologies experts reviewed the source code of a script. They found a fragment allowing remote arbitrary code execution. Any external attacker using this vulnerability could obtain control of the server, read sensitive information, edit and delete data on the application pages, and bring the site down entirely. There was a comment from the developer next to the code, which read simply: "What is this?" Most likely, some other developer left the comment during debugging and everyone else discounted its importance, not considering the code's potential consequences.



Conclusion

It's fair to say that the security of most web applications is still poor. Half of sites contain high-risk vulnerabilities. However, every year we see a steady decrease in the percentage of web applications with severe vulnerabilities. The average number of such vulnerabilities per application has fallen by a third compared to 2018. Another positive trend is that companies are taking security more seriously in not just public-facing web applications, but in their internal ones too. We are hopeful that site security will continue improving this year, with a corresponding drop in successful attacks on web applications.

Achieving and consistently maintaining high security of web applications is not an easy process. There are two ground rules, however:

- **Fix any detected flaws as soon as possible**
- **Make processes automatic wherever possible**

To follow these rules, companies should provide developers with training in secure development methods. Tools for automated source code analysis are a good complement to security analysis of web applications. Together, these will reduce the flaws and vulnerabilities arising in development. We also recommend preventive measures such as a web application firewall (WAF). Since web applications are constantly changing and every new change carries the risk of introducing a new vulnerability, WAFs offer a robust defense. To be effective, a WAF must not only detect and prevent known risks on the levels of application and business logic. It must also detect exploitation of zero-day vulnerabilities, prevent attacks on users, and analyze and correlate events for detection of attack chains.

Client snapshot

38

web applications
analyzed in 2019

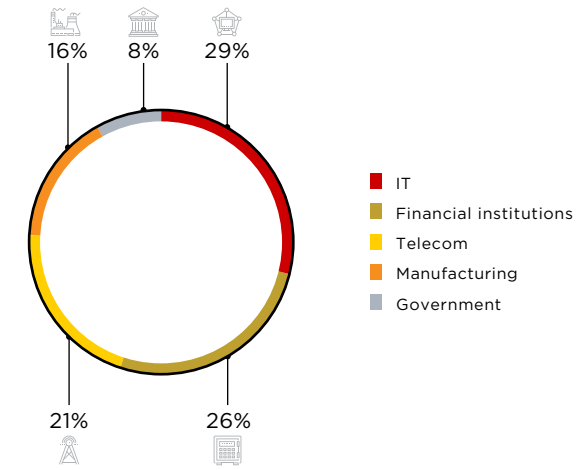


Figure 15. Participant portrait

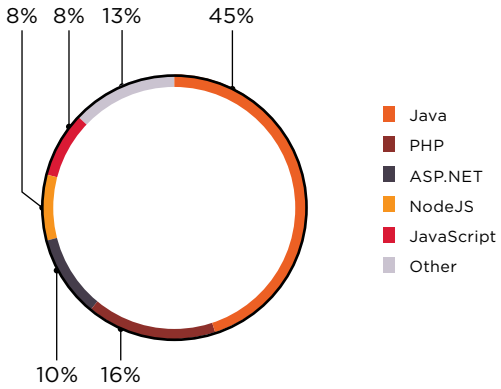


Figure 16. Development tools (percentage of applications)

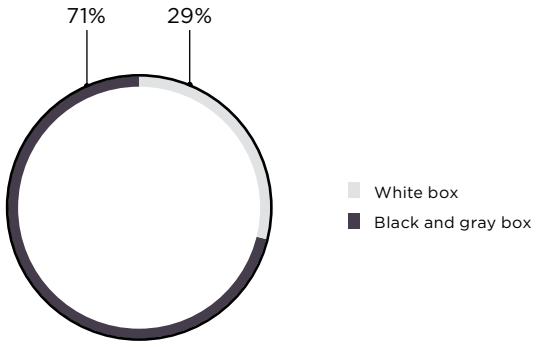
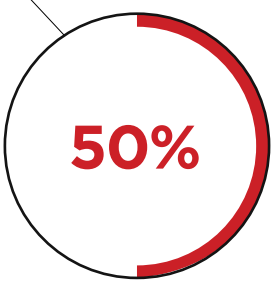


Figure 17. Testing methods (percentage of applications)

of tested web applications
were production systems



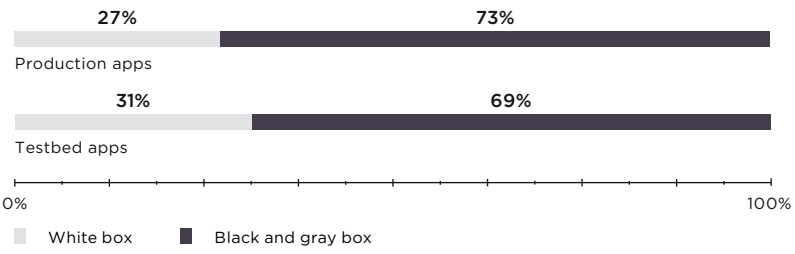


Figure 18. Methods for testing production and testbed systems

Materials and methods

This report includes data from comprehensive security assessments of 38 fully functional web applications for which in-depth analysis with the most comprehensive set of tests was performed in 2019. Results of penetration testing, automated scanning, and testing of e-banking systems are not included here; this information is reviewed in other reports. The dataset does not include applications whose owners did not consent to use of results for research purposes.

The security level of each application was measured manually, using black-, gray-, or white-box methods with the assistance of automated tools. 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 testing refers to security analysis that makes use of all relevant information about the information system, including its source code.

Vulnerabilities were classified according to the industry-standard Common Weakness Enumeration (CWE) system. Because the system is so detailed, for convenience we have focused on vulnerabilities rated in the OWASP Top 10 (2017) and analyzed how frequently we found them in web applications.

The statistics in this report include only code and configuration vulnerabilities. Other common security weaknesses, such as failure to install software updates, are not considered here. Our statistics do not include vulnerabilities related to OWASP Top 10 2017 category A10 (Insufficient Logging & Monitoring), since logging and monitoring practices were out of testing scope. Severity was evaluated based on the Common Vulnerability Scoring System (CVSS v3.1), assigning each vulnerability a rating of Low, Medium, or High.

ptsecurity.com
info@ptsecurity.com

Web-Vulnerabilities-2019_A4.ENG.0003.01.