

A low-angle, upward-looking photograph of several modern skyscrapers with glass facades. The entire image is overlaid with a semi-transparent red filter. The buildings are clustered together, creating a sense of height and urban density. A construction crane is visible at the bottom left, partially obscured by the buildings.

SECURITY TRENDS & VULNERABILITIES REVIEW FINANCIAL SYSTEMS

2017

POSITIVE TECHNOLOGIES

CONTENTS

Introduction.....	3
Executive summary.....	3
1. Research data.....	5
2. Protection flaws.....	6
2.1. Overall statistics	6
2.2. Comparison of in-house and off-the-shelf applications	7
2.3. Comparison of test and production applications	9
3. Vulnerabilities and threats in online banking systems.....	10
Two-factor authentication flaws.....	12
Insufficient protection from brute-force attacks	12
Insufficient authorization	12
Business logic flaws	12
Security threats to online banks	13
4. Vulnerabilities and threats in mobile banking systems.....	13
Two-factor authentication flaws.....	15
Insufficient protection from brute-force attacks	16
Insecure data transfer	16
Insufficient authorization	16
Security threats to mobile banking applications.....	16
5. Vulnerabilities and threats in automated banking systems	17
Conclusions.....	18

INTRODUCTION

Banking services are becoming more accessible to clients every year, using advanced technologies to make payments, transfers, and other transactions convenient like never before. In 2016, these technologies increased in popularity thanks to contactless payment systems: PayPass and payWave were joined by the NFC-based Apple Pay and Google Wallet on smartphones.

But the security of web and mobile banking has lagged behind. These applications harbor all of the vulnerabilities and threats typically encountered in application development (WASC TC v. 2). The difference is that in the case of banking applications, these vulnerabilities can lead to serious consequences— theft, unauthorized access to client data and sensitive bank information, and significant reputational losses.

Data in this report has been taken from 2016 security audits by Positive Technologies of e-banking and automated banking systems. Results for 2016 are compared to 2015. The report offers an overview of the state of protection of banking applications and trends in security. The report also includes recommendations that will help vendors, developers, bank employees, and clients to increase the level of security in development, support, and use of banking applications.

EXECUTIVE SUMMARY

Vulnerabilities are becoming more dangerous. The total number of vulnerabilities in bank applications fell in 2016, but the share of critical vulnerabilities grew by 8%, and medium-severity vulnerabilities by 18%. The most common vulnerabilities related to flaws in mechanisms for identification, authentication, and authorization of users.

"Deployed" doesn't mean "defended." Production systems had an average of twice as many vulnerabilities as those still in development.

In-house development is more secure. Applications developed by outside vendors had an average of twice as many vulnerabilities as applications developed in-house.

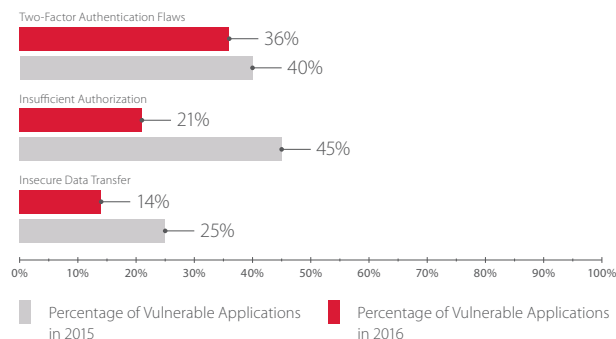
Most online banking applications (71%) contain flaws in their implementation of two-factor authentication. One out of three online banking applications has vulnerabilities that make it possible to steal money.

Mobile banking apps have problems with data storage and transmission. In one out of three apps, an attacker could intercept or bruteforce user credentials. Banking apps on iOS remain more secure than their Android equivalents. But the real problems in protection lurk on the server side: we found dangerous server-side vulnerabilities in every application we tested.

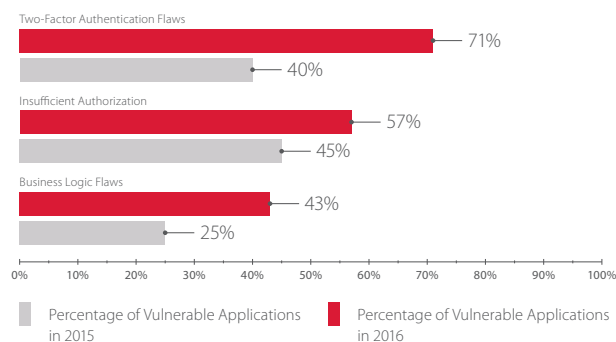
Automated banking systems are usually thought to be out of reach for external attackers. But two thirds of vulnerabilities found in automated banking systems were critical, some of them even allowing administrative server access. In targeted attacks in 2016, attackers tended to make more active use of such vulnerabilities against financial services companies.

2016 saw a smaller number of vulnerabilities identified in financial applications.

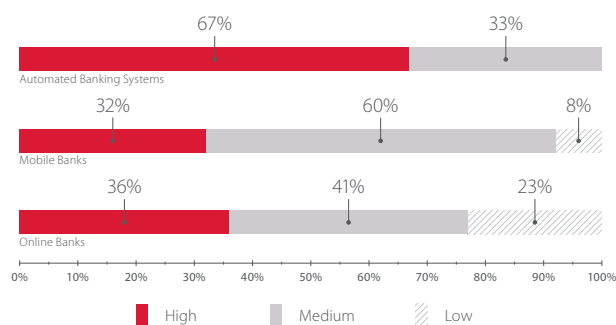
The most common vulnerabilities relate to flaws in identification, authentication, and authorization.



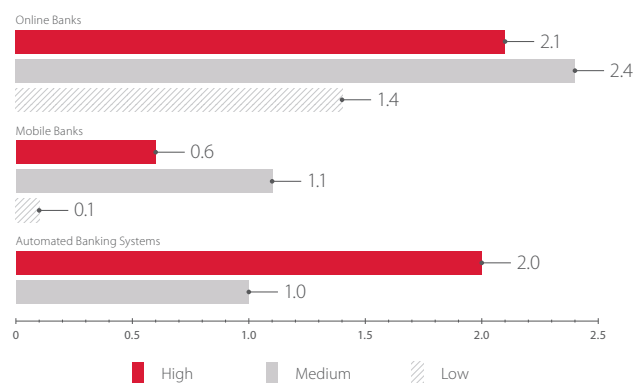
Critical vulnerabilities in mobile banks



Critical vulnerabilities in online banks



Vulnerabilities of different severity levels



Average number of different severity vulnerabilities per application

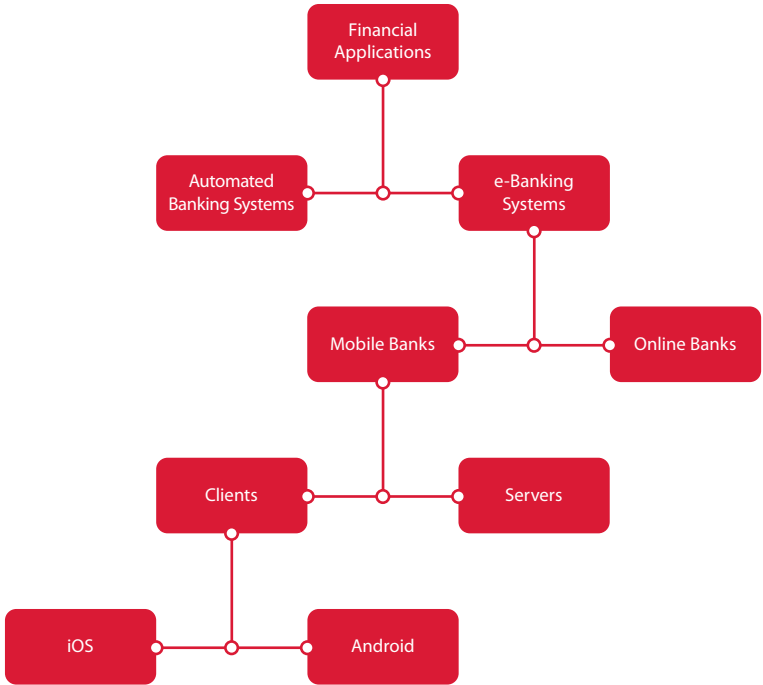
Vulnerabilities increased significantly in severity.

Online banking applications tend to contain more vulnerabilities than mobile banking apps and automated banking systems.

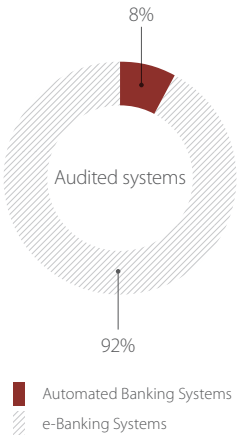
1. RESEARCH DATA

This report encompasses work performed on 24 applications used for financial transactions and automation in the banking industry. Data comes from audits performed by Positive Technologies during 2016.

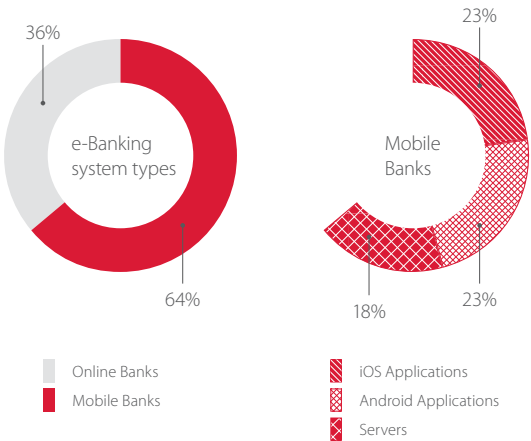
Audited systems in 2016 included online banking applications, mobile banking applications, and automated banking systems.



Two thirds of the audited e-banking systems were mobile banking applications, which included both Android/iOS clients and server-side software.



All audited e-banking applications were intended for use by retail (individual) customers.

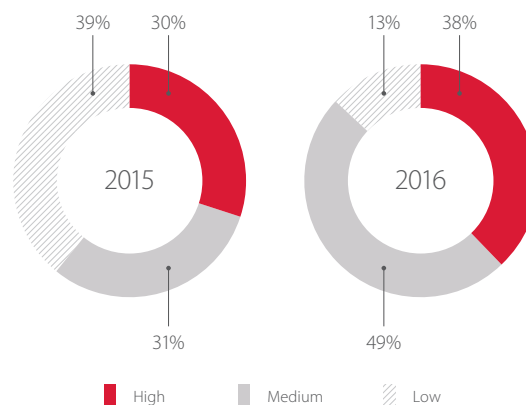


2. PROTECTION FLAWS

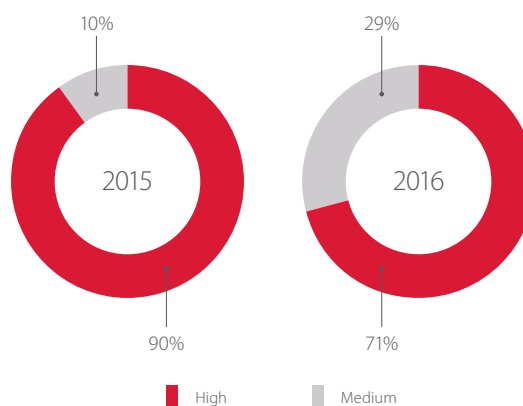
2.1. Overall statistics

Every application tested had multiple security flaws. On average, each financial application audited in 2016 contained six vulnerabilities, which is much lower (by three) than in the previous year. This improvement reflects efforts by banks to tackle security issues head-on.

High-severity vulnerabilities grew by 8% in 2016. Medium-severity vulnerabilities increased by 16%.



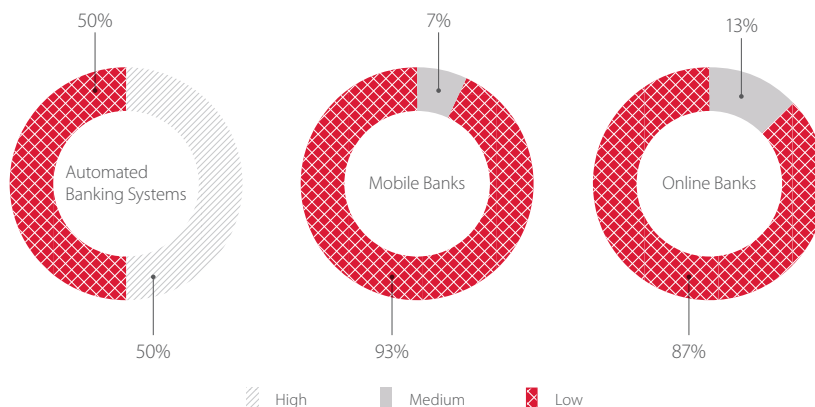
Vulnerabilities by different severity levels



Most severe vulnerability found (% of applications tested)

In 2015, only a third of vulnerabilities were of high severity, but they were distributed more evenly among the audited applications; only 10% of applications had no critical vulnerabilities. However, in 2016 we see a different picture: 38% of vulnerabilities were critical, with these concentrated in 71% of the applications tested. One out of three applications had only medium- or low-severity vulnerabilities.

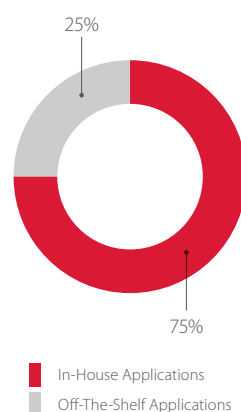
Financial applications still remain at risk.



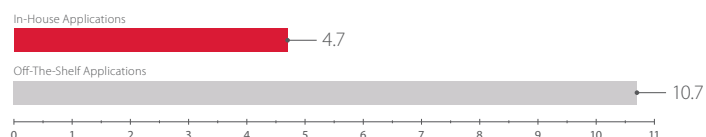
State of security (% of applications tested)

2.2. Comparison of in-house and off-the-shelf applications

75% of tested applications were developed by financial services companies in-house.



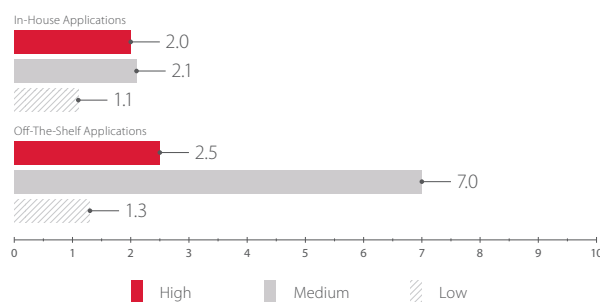
Shares of in-house vs. off-the-shelf applications



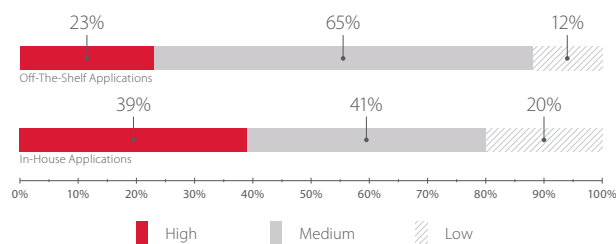
Average number of vulnerabilities per application

The results of testing for off-the-shelf applications still disappoint. These applications contained twice as many flaws as their in-house counterparts.

In off-the-shelf applications, 23% of vulnerabilities were of high severity, mainly XML External Entity and Business Logic Flaws exploitation. 65% of vulnerabilities were of medium severity. Among in-house applications, 39% contained critical vulnerabilities, most of which related to insufficient authorization and implementation of two-factor authentication.

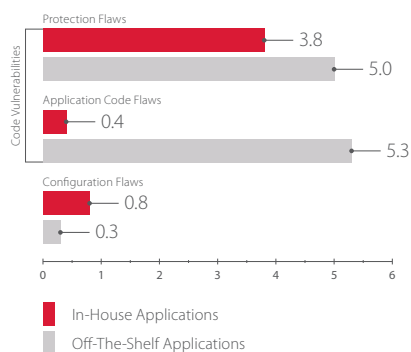


Average number of vulnerabilities of different severities in a single application

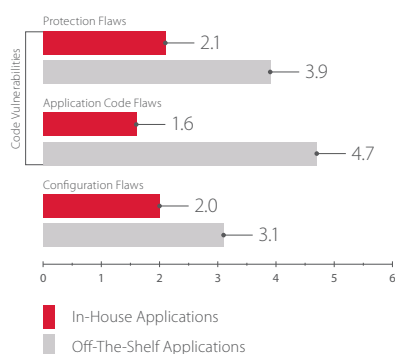


Vulnerabilities of different severity grades

In-house applications tended to have flaws in implementation of protection mechanisms, but their code contained fewer bugs and vulnerabilities than off-the-shelf ones. Flaws in protection mechanisms are code vulnerabilities as well, of course, but these flaws do not arise during development per se. Programmers are not to blame, since they are writing code based on the articulated requirements. Rather, such flaws arise during the design stage due to failure to take into account the nuances of protection, authentication, and authorization.



Average number of vulnerabilities per application (2016)



Average number of vulnerabilities per application (2015)

The findings show that in-house development creates more secure results. So the cost of hiring programmers and setting up a Secure Software Development Lifecycle is justified by security outcomes.

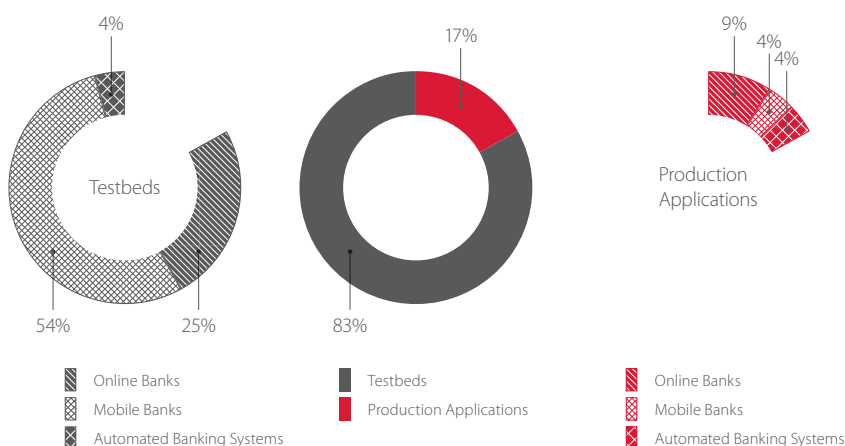
Another argument in favor of in-house development is the lengthy vulnerability remediation process of outside vendors. What happens if a vendor is reported about a vulnerability in an off-the-shelf application used by a client? First a vulnerability is discovered by the client or a security researcher, then this information is sent to the vendor. The vendor tries to understand the true cause of the vulnerability and verify that the vulnerability actually does, in fact, originate in the vendor's product. Then the vendor's developers create a patch, test it on all versions of the product, and make it available to clients. The client must then test it on their own systems and, if all goes well, install it on production systems.

Depending on an outside vendor for security patches has both plusses and minuses: theoretically, the client doesn't need to spend money on auditing the application and can simply install vendor updates as they are released. By the same token, an attacker who finds a vulnerability in an application can use and re-use the same vulnerability against multiple banks that use applications developed by the same vendor.

By contrast, banks with in-house development can fix vulnerabilities much more quickly. Speed is often the name of the game when reacting quickly to prevent financial losses and reputational damage.

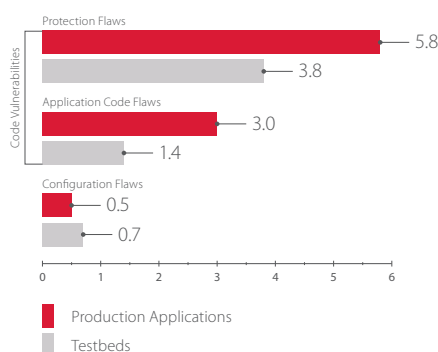
2.3. Comparison of test and production applications

One of the key steps for having a well-protected application is to test it before making it public. 83% of the audited applications were in development, but the average number of vulnerabilities in them was smaller than in production applications.

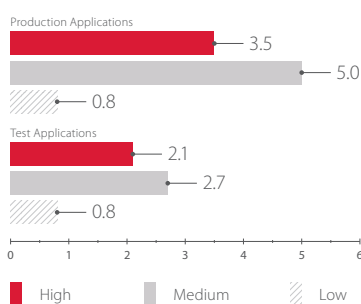


Test vs. production applications

Production systems had almost twice as many vulnerabilities as test systems.



Average number of vulnerabilities of various categories in test and production systems



Average number of vulnerabilities of various severity in test and production systems

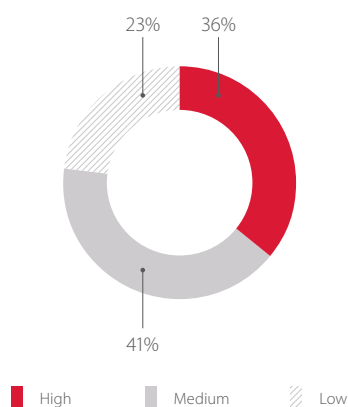
The largest number of vulnerabilities, on both test and production systems, related to flaws in implementation of protection mechanisms. Financial applications contained a rather large number of vulnerabilities, especially Cross-Site Scripting and XML External Entity. This may be because audits tended to concentrate on applications with recent new functionality. Code changes can make new vulnerabilities occur, making it important to perform regular security audits.

The source code of a web application can contain many critical vulnerabilities as well. One way to minimize such vulnerabilities is to practice the Secure Software Development Lifecycle (SSDLC). For timely identification of vulnerabilities in code, regular audits of code quality are required, such as with white-box testing (including use of automated analysis tools).

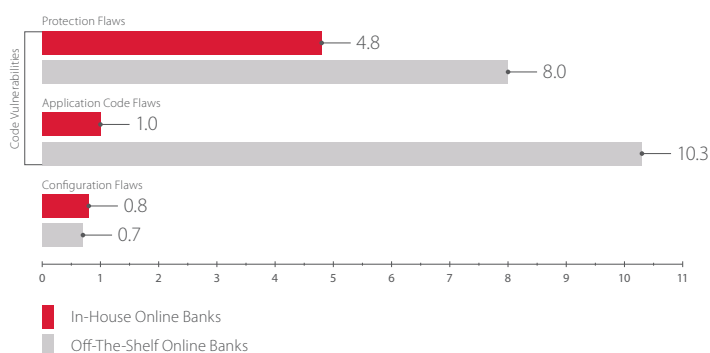
3. VULNERABILITIES AND THREATS IN ONLINE BANKING SYSTEMS

All online banking applications (except for one) contained at least one critical vulnerability. Online banking applications contained an average of 2.1 high-severity vulnerabilities, which is a drop of 50% compared to the prior year (4.2).

100% of online banks had flaws related to implementation of protection mechanisms.

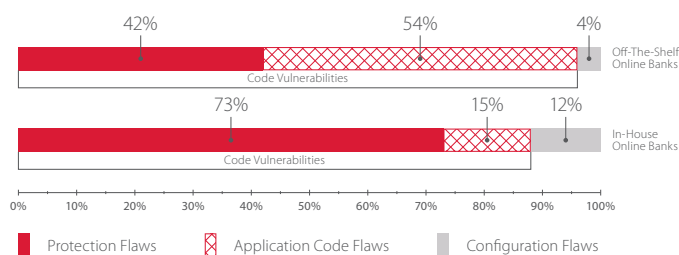


Severity of vulnerabilities detected



Average number of various vulnerabilities in financial web applications

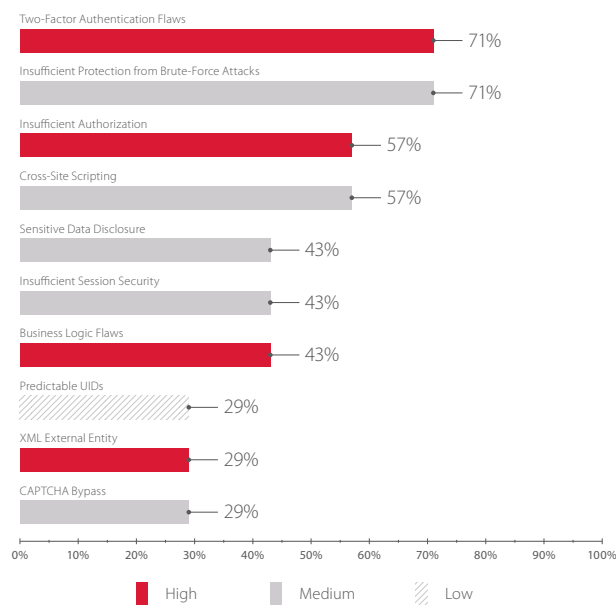
Online banking applications developed in-house tended to have problems in implementation of protection mechanisms (such as insufficient protection from bruteforcing, weak password policy, or bypassable CAPTCHA). By contrast, off-the-shelf applications contained more code vulnerabilities (for example, Cross-Site Scripting and XML External Entity).



Vulnerabilities by category

In addition, half of in-house online banking applications use out-of-date software. One of the tested applications contains PrimeFaces 5.3.x, an out-of-date framework with a vulnerability that enables an attacker to upload any file (including a web command-line interpreter) to the server in order to perform privilege escalation attacks. If the application has excessive privileges, or if the attacker exploits OS vulnerabilities, this can result in complete control over the server.

The list of most common vulnerabilities is similar to last year's. The same vulnerabilities keep recurring, the difference being that in 2016, the top spots in the list went to high-severity vulnerabilities, as opposed to low-severity ones in prior years.

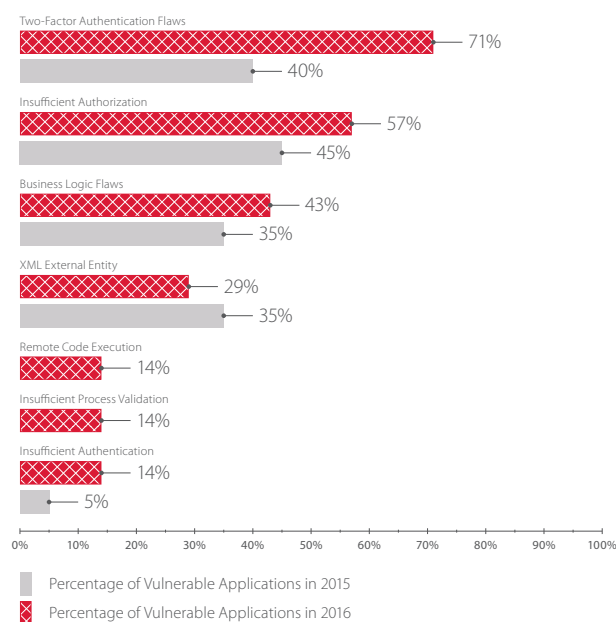


Top 10 online bank vulnerabilities

All online banking applications contained medium-severity vulnerabilities. Insufficient Session Security and Cross-Site Scripting make bank clients vulnerable to attacks (such as interception of cookies values or theft of user credentials). These two vulnerability types have now made the top 10 list of vulnerabilities in 2016 for all web applications, and not only for online banking applications, as was shown in our report on vulnerabilities in web applications in 2016.¹

Incorrect CAPTCHA implementation (for example, an application allowing re-use of a CAPTCHA session) makes such measures effectively useless and enables attacks that aim to bruteforce user credentials.

2016 was marked by growth in critical vulnerabilities related to flawed authentication mechanisms. This was particularly evident in online banking applications, 71% of which were found to have flaws in implementation of two-factor authentication.



Critical vulnerabilities of online banks

¹ <https://www.ptsecurity.com/ww-en/analytics/>

Two-factor authentication flaws

In 2015, we already saw this problem become more prevalent among all authentication flaws, but at the time, only 33% of off-the-shelf and 45% of in-house banking applications were vulnerable.



Two-factor authentication for account access was present in only 71% of the tested applications. Only half required a one-time password for confirming transactions.

Flaws in two-factor authentication (other than its absence) include:

- + Client-side generation of one-time passwords
- + Failure to associate a one-time password with the operation being performed
- + Unlimited number of attempts to enter a one-time password
- + Unlimited one-time password lifetime

If a second authentication factor (in online banking, this is usually a one-time code sent to the user's mobile phone) is missing or implemented incorrectly, attackers can access a user's account and conduct financial transactions simply by obtaining a user name and password.

Insufficient protection from brute-force attacks

More than half of financial web applications fail to provide sufficient protection from brute-force attacks on user names and passwords. Specific examples of such weaknesses include:

- + Use of predictable IDs, such as mobile number
- + Use of default, hard-coded, or predictable passwords, such as based on the user's birthdate
- + Bypassable or absent CAPTCHA mechanism
- + No lockout after excessive failed login attempts

Safeguarding user authentication data must be an absolute priority for banks.



25% of tested online banking applications contained two particular kinds of vulnerabilities at the same time: poor protection from bruteforcing of user accounts, and flawed implementation of two-factor authentication. This means that for almost one out of three online banking applications an attacker could gain total control over client accounts.

Insufficient authorization

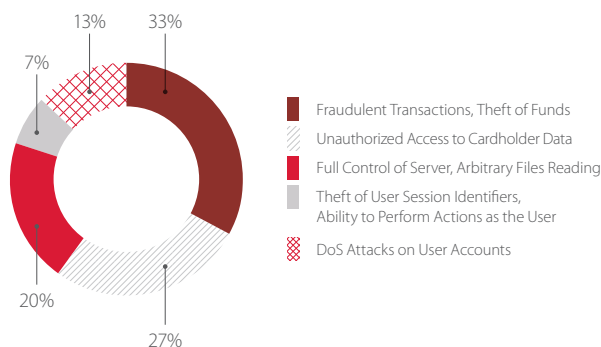
As before, many applications fail to correctly implement authorization mechanisms and restrictions on access to sensitive information (57%). Using this kind of vulnerability, an attacker could target bank clients and obtain unauthorized access to sensitive banking information. In the case of one online bank, an attacker could learn the loan repayment schedule of a client; at another, an attacker could learn the account balances of other users.

Business logic flaws

Due to flaws made during the design stage, one out of three online banking applications is vulnerable to logic-based attacks. One such attack takes advantage of rounding, transferring 1.0009 units of currency from one account to another. The transfer amount is rounded by the bank's system and as a result the recipient's account is credited with 1.001 units. While this may not seem like much profit, this transaction can be repeated many times.

Security threats to online banks

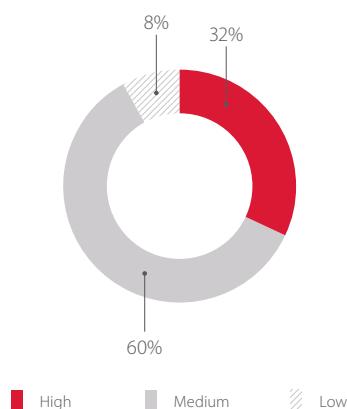
Vulnerabilities in web applications create large reputational and financial risks for banks and their clients. Security gaps can cause very tangible impacts, such as in the 33% of applications vulnerable to fraud. In 27% of applications, an attacker could access sensitive client information, such as account balances or loan payment schedules.



Potential impact of attacks on online banks

4. VULNERABILITIES AND THREATS IN MOBILE BANKING SYSTEMS

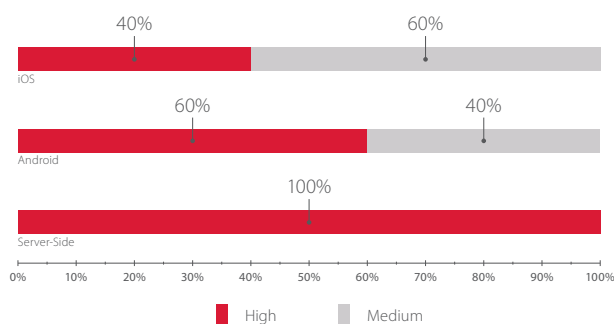
64% of mobile banking applications contained at least one critical vulnerability. Applications contained an average of 0.9 high-severity vulnerabilities, which is less than found for the tested online banks.



Severity of vulnerabilities found

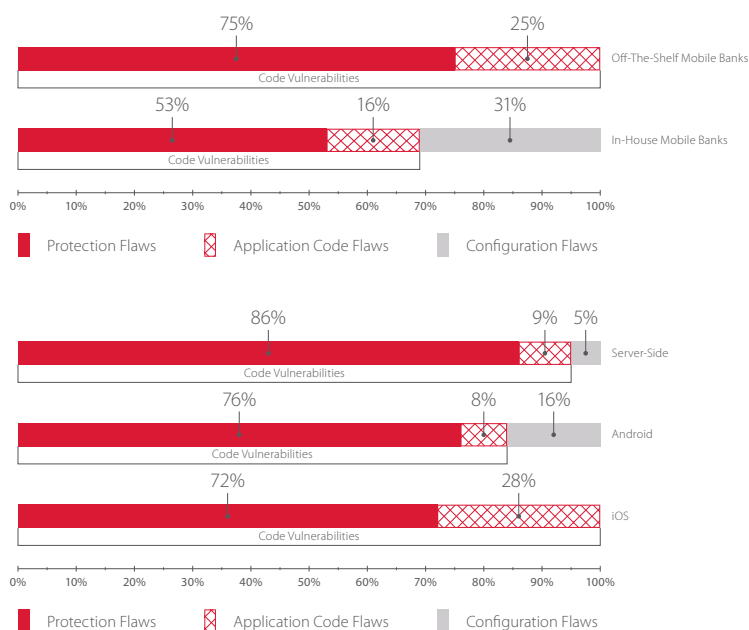
iOS apps remain more secure than their Android counterparts.

Server-side components of mobile banking applications were most likely to harbor critical vulnerabilities.



Most severe vulnerability found (% of mobile banking applications)

For all mobile banking applications, we analyzed two versions: one for Android, the other for iOS. In some cases, the iOS application did not contain vulnerabilities found in the equivalent Android application.

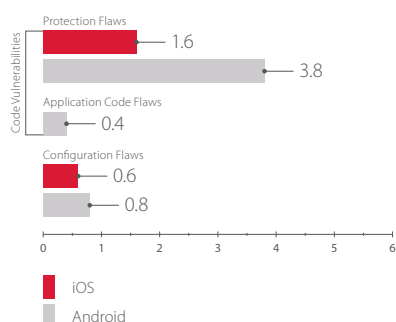


Vulnerabilities by category

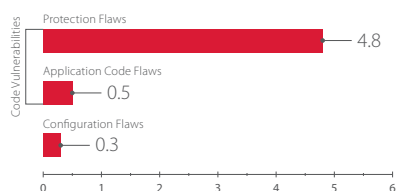
All server portions of mobile banking applications contained critical vulnerabilities. Each contained vulnerabilities related to insufficient authorization and authentication (including two-factor).

Notably, off-the-shelf mobile applications did not contain any vulnerabilities related to configuration flaws.

On average, iOS applications contain fewer vulnerabilities than other systems.



Average number of vulnerabilities of various categories (mobile banking applications)

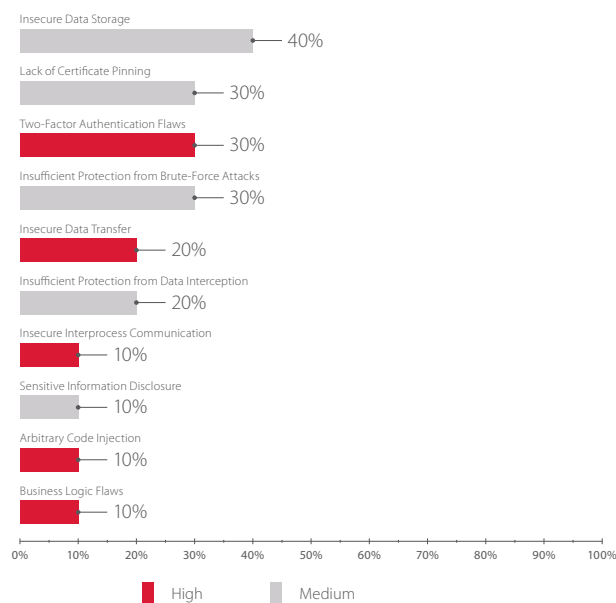


Average number of vulnerabilities on mobile bank server side

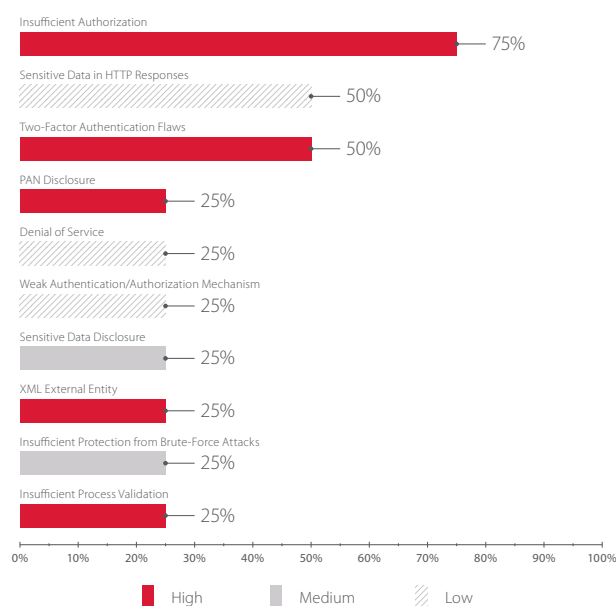
As stated above, flaws in protection mechanisms occur at the code level, but since these flaws originate with design errors instead of programming errors, they are considered separately.

Such vulnerabilities as Insufficient Protection from Data Interception and Insecure Data Transfer affected only in-house mobile banking applications.

Vulnerabilities related to insecure data storage and/or insecure data transfer were often detected in client mobile applications, just as in 2015. These two types of vulnerabilities can be of either high or medium severity depending on the exploitation context.



Top 10 vulnerabilities on the client side of mobile banking systems



Top 10 vulnerabilities on the server side of mobile applications

Lack of certificate pinning (which validates server identity) allows attackers to intercept and modify transmitted data. With one bank, this vulnerability could be used for compromising one-time passwords, while in another case, an attacker could intercept the user's credentials and access the account.

The most acute security issue for the server side of mobile applications is insufficient authorization. This vulnerability allows an attacker to gain access to sensitive information stored on the server.

Two-factor authentication flaws

30% of iOS and Android mobile applications contained flaws in two-factor authentication. The two-factor model was the same in all the applications tested, consisting of:

1. Knowledge (user name and password)
2. Possession (smartphone to which the one-time password is sent).



For user convenience, some application developers limit the functionality of a mobile bank, while simplifying the authentication process.

In one mobile application, a combination of "mobile phone number + date of birth" was used as the user ID. This information is not secret and can be obtained by attackers from public sources such as social networks. Even worse, one's date of birth and phone number cannot be changed if compromised, unlike a traditional user name and password. In this case, two-factor authentication is degraded to one-factor authentication, and its security depends only on a one-time password, which does not provide adequate protection under the following conditions:

- + The phone receiving the one-time password is lost or stolen.
- + The attacker knows the victim and has physical access (even for a few minutes) to the phone.
- + The one-time password is not tied to a specific operation. Attackers with access to the application, or with control over the base station to which a legitimate user is connected, can generate multiple one-time passwords for future use by modifying the time on the user's smartphone.

Insufficient protection from brute-force attacks

To access a mobile banking app, it is often enough to enter a four-digit code. If there is no limit on the number of login attempts, attackers can quickly try all 10,000 possible combinations using special software.

In one application, an attacker could infer the validity of an entered user name based on the server response time: the response would be faster or slower depending on whether the user name was valid.

In another vulnerability related to insufficient protection from brute-force attacks, the server response indicated only whether the entered PIN code matches the code stored on the server. The server did not return a new session ID; instead, the client application used an ID stored on the device. So the client application determined whether the user entered the correct PIN code based only on the server response, even though server responses can be easily faked.

Insecure data transfer

Mobile apps most often use the secure HTTPS protocol for server communication. But in some cases, apps had issues with client-side implementation of functionality for establishing an encrypted connection. For example, one app allowed the server to use an untrusted self-signed certificate. In a man-in-the-middle attack, it would be possible for an intruder to spoof the server's certificate and intercept (or modify) data during transmission, including authentication data. It would not be necessary to install certificates on the victim device; all an attacker would need is to control the data transmission channel.

Insufficient authorization

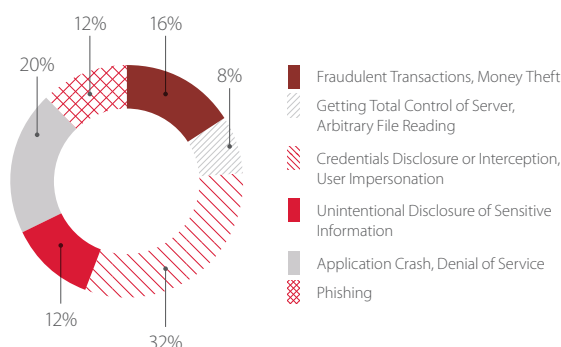
In one of the mobile apps we analyzed, an unfortunate error was made during development: any attacker who obtained (or stole) a smartphone, even without the user name or password of the client of a mobile bank, could still obtain access to minimal app functionality, including access to information about bank cards, payments, and auto-pay settings.

On the screen for entering the PIN code, the attacker could simply press the Back button, and then be taken to the user name/password authentication screen for the mobile bank. But the current user remained logged in, with information available as described above.

Security threats to mobile banking applications

32% of the tested mobile banking applications allowed attackers to decrypt, intercept, and brute-force credentials, or bypass the authentication process entirely. Due to this, attackers could log in to the mobile application as a legitimate user and perform transactions.

Attackers with access to the application, or with control over the base station to which a legitimate user is connected, can generate multiple one-time passwords for future use by modifying the time on the user's smartphone. Attackers can forge transactions and use an intercepted one-time code for transactions of up to €5,000 (depending on bank limits) by using Cross-Site Scripting or an MITM attack.



Potential impact of attacks on mobile banking applications



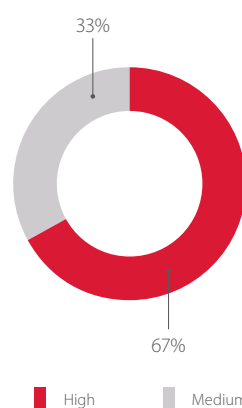
Attacks on mobile bank applications in 2016 could cause serious harm to both banks and their clients, since most attacks involved:

- + Fraud
- + Obtaining control over the server side of the banking application
- + Impersonation of a legitimate user

5. VULNERABILITIES AND THREATS IN AUTOMATED BANKING SYSTEMS

Automated banking systems are a special class of financial applications supporting banking operations as a whole: cash management and customer service (such as opening accounts and issuing loans), foreign exchange and money market operations, securities transactions, and more.

Two thirds of vulnerabilities found in automated banking systems were critical; the remaining ones were classified as being of medium severity.



Severity of vulnerabilities found

In 2016, the following security flaws were among the more typical critical vulnerabilities in these systems:

- + Insufficient Authorization
- + Insufficient Authentication
- + XML External Entity

When testing one automated banking system, our security experts gained administrative access to the bank's server, meaning that a potential attacker could conduct fraudulent transactions and remain unnoticed. The possibilities for such fraudulent transactions are practically limitless: attackers could create new accounts and set their balance, or create counterfeit payment orders to other institutions. Most likely, this attack would be detected only when the amount being transferred out exceeds the balance of the bank's correspondent account.

For example, Bangladesh bank suffered major losses in early 2016 from attacks on their automated banking systems, totaling over USD \$81 million.²



In our analysis of 2016 incidents, we note that targeted attacks against banks were often aimed at changing the recipient of payment orders. This attack vector is difficult to implement, since automated banking systems are closed off to outsiders, but can have catastrophic consequences.

CONCLUSIONS

The security level of financial applications remains low, as shown by our research. In our audits of financial applications in 2016, we found fewer vulnerabilities, but of higher severity, compared to the prior period.

Poor implementation of protection mechanisms is the cause of the most flagrant security issues found in online and mobile banking applications. By and large, these vulnerabilities can be avoided before the first line of code is ever written—proper architecture and careful formulation of technical requirements should account for the subtleties of implementation of (two-factor) authentication and authorization mechanisms.

Vulnerabilities in source code can be avoided at the development stage. Secure Software Development Lifecycle (SSDLC) practices and careful testing of protection mechanisms ensure a more robust and secure code base.

To mitigate risks, we urge banks to assess security at all software stages from development to use by real clients, with timely action to address vulnerabilities. Experience has proven that the most effective method to detect web application vulnerabilities is auditing source code with the help of automated analysis.

² <https://www.wired.com/2016/05/insane-81m-bangladesh-bank-heist-heres-know/>

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.

© 2017 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.