Vulnerabilities and threats in mobile banking

ptsecurity.com

Contents

About the research

7 **banks**
Android clients
iOS clients
servers

In 2019, we chose 14 fully featured mobile banking applications (client + server) for our research. Criteria for inclusion were:

- Both Android and iOS clients were available and analyzed for the bank in question.
- The bank application had been downloaded from official app stores (Google Play and Apple's App Store) more than 500,000 times.
- System owners agreed to use of security assessment results for research purposes.

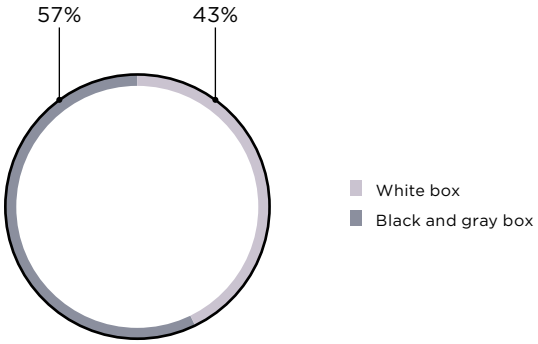

Figure 1. Testing methods used (percentage of applications)

The security level of each application was assessed 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 knowledge of the application. Gray-box testing is similar to black-box testing, except that the attacker is a user who has some privileges. White-box testing refers to security analysis that makes use of all relevant information about the application, including its source code.

This report summarizes client- and server-side vulnerabilities in mobile banking applications related to faults in application code, client–server interaction, and implementation of security mechanisms. Other common security weaknesses, such as failure to manage software updates, are not considered here. Vulnerability risk was assessed based on the impact of a potential attack on user data and the application itself, taking feasibility into account. When grading each vulnerability, we made a qualitative assessment and assigned high, medium, or low risk.

Executive summary

None of the tested mobile banking applications has an acceptable level of protection.

Client side

The client side refers to a mobile banking application installed on the user's device.

- In 13 out of 14 applications, attackers can access user data from the client side.
- 76 percent of mobile banking vulnerabilities can be exploited without physical access to the device.
- More than a third of vulnerabilities can be exploited without administrator (jailbreak or root) rights.

Server side

The server side is a web application that interacts with the mobile client over the Internet by means of a special application programming interface (API).

- Server sides contain 54 percent of all vulnerabilities we found.
- On average, each mobile bank has 23 server-side vulnerabilities.
- Half of mobile banks are vulnerable to fraud and theft of funds.
- At five out of seven banks, hackers can steal user credentials. At one third of banks, card information is at risk.

Client-side vulnerabilities

None of the tested clients has an acceptable level of protection

iOS client applications contain fewer vulnerabilities than their Android counterparts. No flaws in iOS banking apps were worse than "medium" in severity. By comparison, 29 percent of Android apps contain high-risk vulnerabilities.

Figure 2. Level of protection of client applications (number of applications)

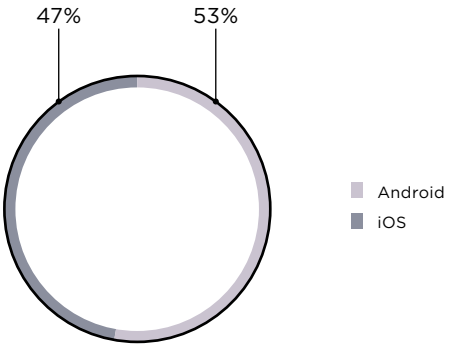

Figure 3. Percentage of vulnerabilities in Android vs. iOS clients

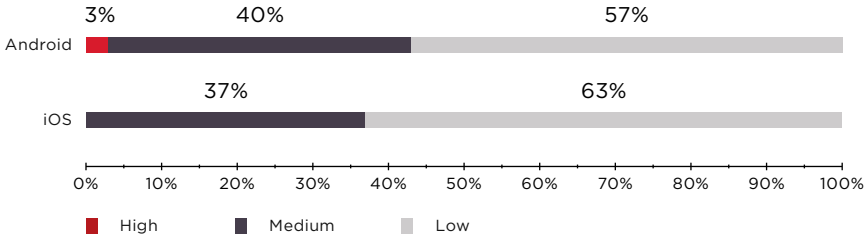

Figure 4. Vulnerabilities by severity

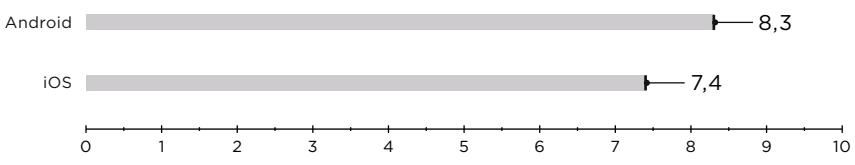

Figure 5. Average number of vulnerabilities per application

More features, more risks

The most dangerous vulnerabilities we found are in Android applications and involve insecure deeplink handling. Deep linking is used differently on iOS and Android: developers on Android have more freedom of implementation. This explains the larger number of vulnerabilities in Android applications compared to iOS. However, this does not mean that iOS developers are immune. Mobile banking security depends above all on a Secure Software Development Lifecycle (SSDL).

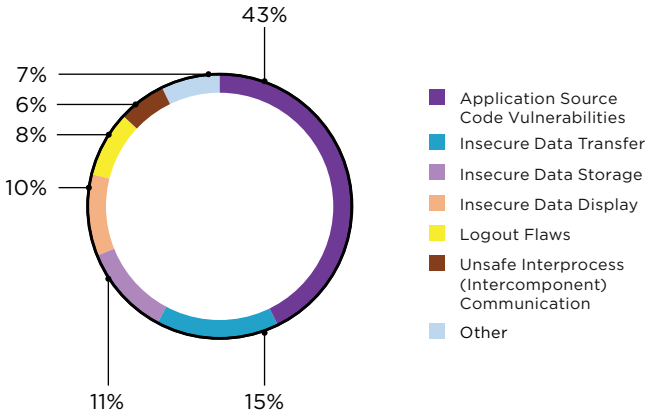

Figure 6. Vulnerabilities by type

100%

of mobile banking clients contain vulnerabilities in their code. For example:

- Code is not obfuscated.
- Protection against code injection and repackaging is absent.
- Code contains names of classes and methods.

Our research demonstrates that insufficient code protection leaves banks vulnerable to source code analysis. To exploit vulnerabilities in code, all attackers need is to download the application from Google Play or the App Store and then decompile it.

Lack of obfuscation allows attackers to analyze the code and find important data, such as:

- Testing-related usernames and passwords
- Encryption keys and parameters from which keys can be derived
- Salts for hashing and encryption

Attackers can then use this information to obtain credentials and access web servers. What's more, hackers can analyze the application algorithm and exploit flaws in business logic. Competitors may also want to know how the application is designed in order to copy new features for their own products.

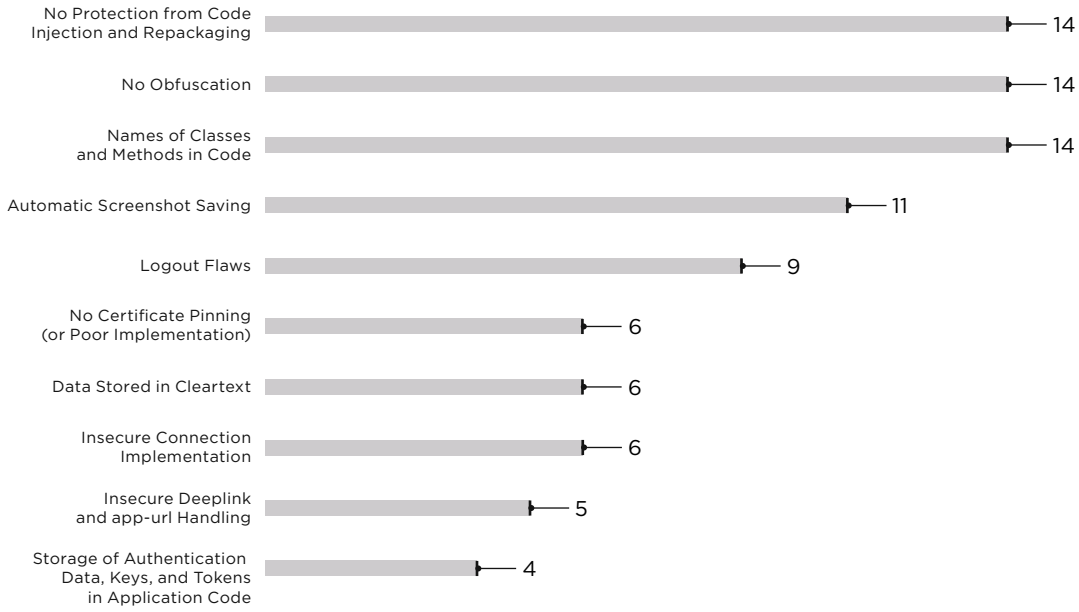

Figure 7. Top 10 mobile banking vulnerabilities (number of applications affected)

Tip for developers

Use obfuscation to make it difficult for attackers to read and analyze code. One example of code obfuscation is to remove characters at compile time. Names of classes and methods in the source are replaced by random or single-letter names. Developers can use special software, such as ProGuard for Android or Sirius Obfuscator and SwiftShield for iOS.

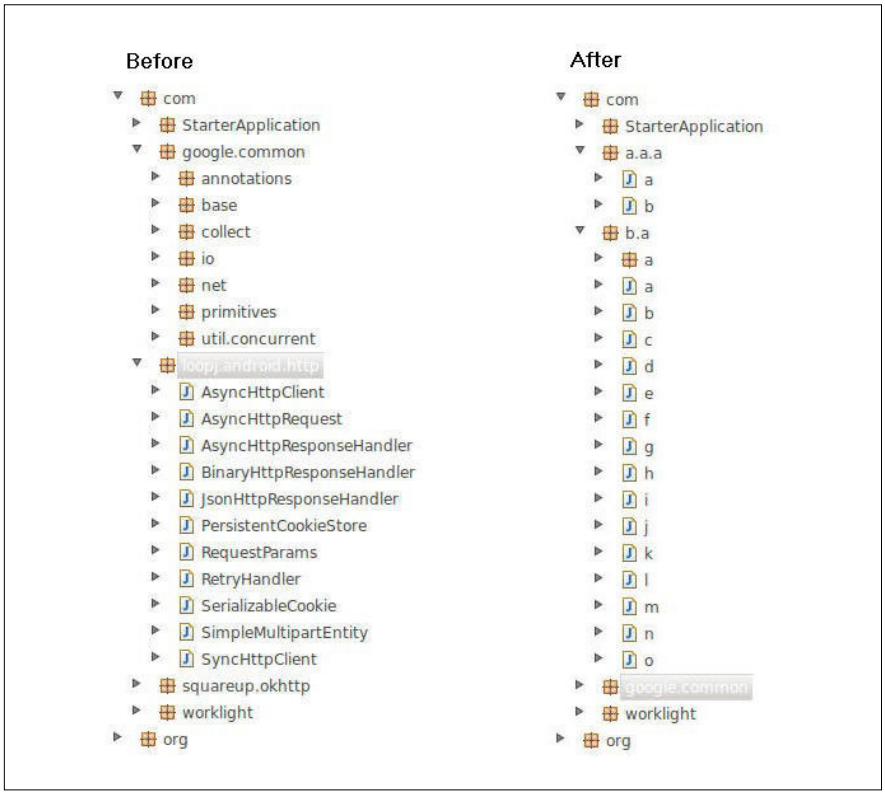

Figure 8. Decompiled executable files before and after character removal

Figure 9. Prerequisites for vulnerability exploitation

67%

of attacks against individuals in Q4 2019 involved social engineering

To exploit some client-side vulnerabilities, all an attacker would need to do is convince the victim to install a malicious app, perhaps with the help of phishing.

Insecure deeplink handling is a critical vulnerability with the potential to cause financial losses for banks. For example, one banking application failed to filter deep linking URLs. The problem is that embedded WebView components can load arbitrary links. So attackers could take advantage of this by loading a link to a web page containing malicious code and interact with the JavaScript interfaces available in those WebView components. Positive Technologies experts developed test scripts and demonstrated SMS interception. They were able to obtain card numbers by manipulating the ability to scan bank cards using the on-board camera or via NFC. Attackers can display a malicious page in the application's interface and prompt to scan a card. For the user, everything looks like a regular banking transaction, except that criminals (and not the bank) will be the ones receiving the data.

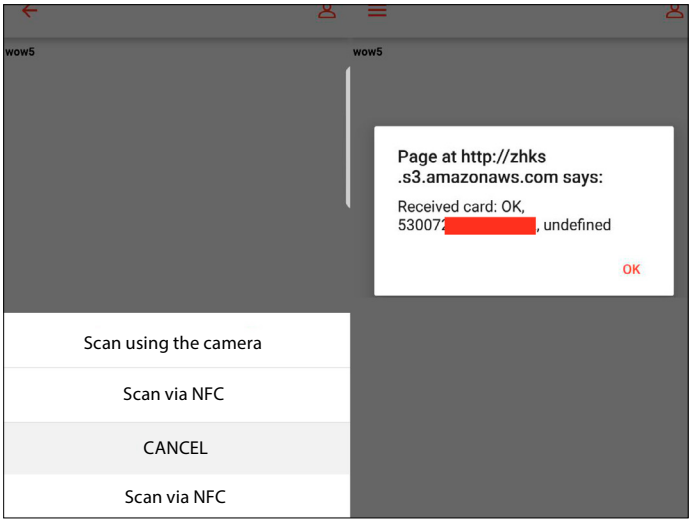

Figure 10. Card scan manipulation

Deep linking is a technology that allows users to navigate between applications (or sections within an application) to a specific location using special links, similar to hyperlinks in web applications.

Tip for developers

Deep linking creates another point of entry for attackers. Remember that all parameters passed using deep linking come from an insecure source, so verify and filter them before passing them to source code methods.

11 out of 14 mobile banks allow automatic screenshot capture, a feature that helps to quickly view recently used programs. But screenshots may contain sensitive data such as card information and account balances.

The client-side file system of almost half of applications contains unencrypted sensitive information. To access this data, attackers need root or jailbreak rights. Rooting or jailbreaking the device can be done with physical access or remotely by means of malware. In one mobile banking application, our experts found card balance statements stored on the phone. Another application went so far as to save the user's PIN code, allowing attackers to access the user's account.

43%

of applications store important data on the phone in cleartext

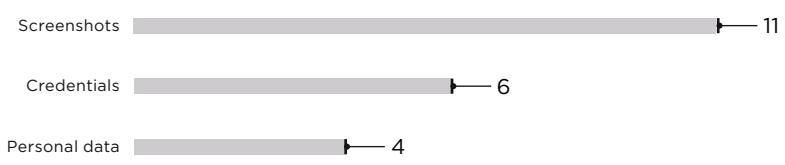

Figure 11. Disclosed information (number of applications affected)

Tip for developers

Store as little data on user devices as possible. Request data from the server only as needed by the application and delete it when finished. Encrypt sensitive information stored on the device and ensure that encryption keys are securely managed. To protect data from screenshots, use a special background image to block out app screens containing sensitive information.

Only one of the tested mobile banks did not contain vulnerabilities allowing attackers to access user data. 13 out of 14 applications were vulnerable to man-in-the-middle attacks due to a lack of certificate pinning to validate SSL certificates, issues with connection implementation, and use of insecure external object references. If successful, attackers can access sensitive user data, as well as read and tamper with data transferred between the server and the client application.

Figure 12. Top three mobile banking threats (number of applications affected)

Server-side application vulnerabilities

More than half of mobile banks contain high-risk server-side vulnerabilities. Overall, not a single server side had a security level better than "medium." Three had a security level that was "low," and one "extremely low."

23
server-side vulnerabilities
exist in each mobile bank on average

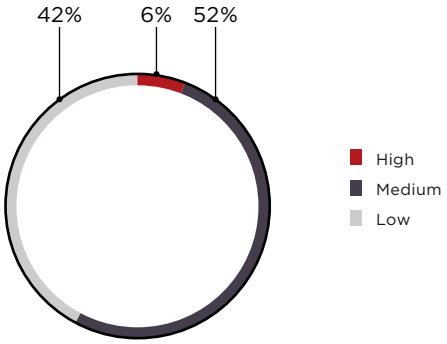

Figure 13. Vulnerabilities by severity

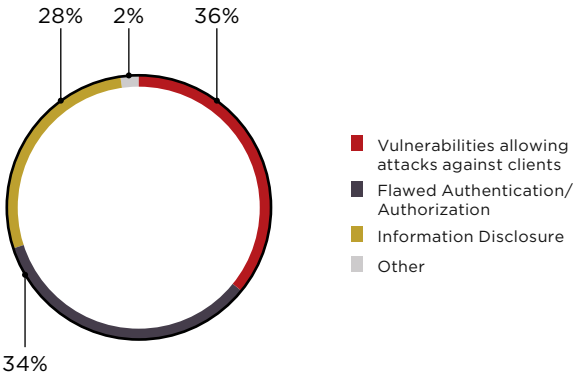

Figure 14. Vulnerabilities by type

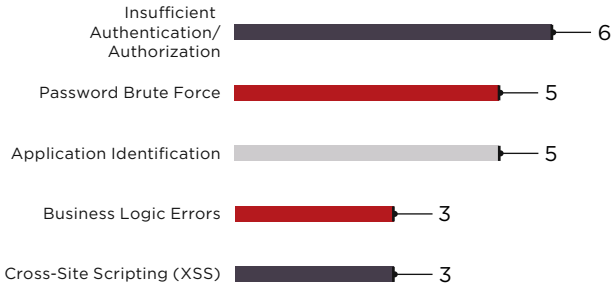

Figure 15. Top five server-side vulnerabilities (number of servers affected)

Most bruteforce vulnerabilities are caused by flaws in the one-time password (OTP) mechanism. The most common problem is that a password remains valid even if the number of password input attempts is exceeded. Attackers can access the user's account and take advantage of OTP flaws to impersonate the user in various transactions, including transferring funds.

Three out of seven mobile banks contain server-side vulnerabilities in business logic. In most cases, these vulnerabilities impact functionality directly useful for fraud attempts. Business logic errors may cause significant losses to banks and even lead to legal complications.

Tip for developers

Apply SSDL and integrate assessment of code security as early as possible in the development process.

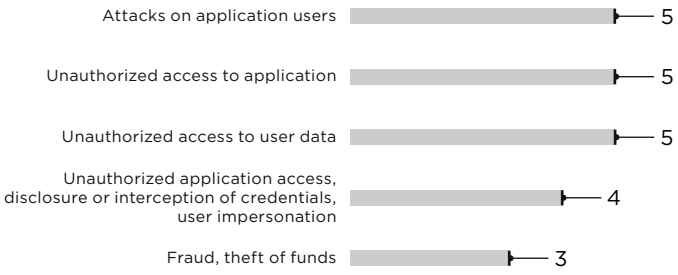

Figure 16. Mobile banking threats (number of servers affected)

Card information is at risk at **two out of seven mobile banks**

Five out of seven mobile banks have server-side vulnerabilities that hackers can exploit against users. For example, insufficient extension checking of uploaded files in one mobile application allows attackers to upload malicious executable files to the server. If a bank employee ran such a file, a malicious script could run and steal data from the server, for example.

Unauthorized access to applications usually results from authentication and authorization flaws. For example, attackers can bruteforce a user's password during authentication and access the victim's account. Next, if attackers succeed in bypassing one-time password protection by exploiting OTP flaws, they can impersonate the victim.

User credentials proved to be the most vulnerable prey: mobile banking usernames and passwords are jeopardized on the server side of five mobile banks. Personal data can fall into the hands of attackers in more than half of the mobile apps. This information may include usernames, account balances, transfer confirmations, card limits, and the phone number associated with a victim's card.

When exploited, mobile banking vulnerabilities yield information that can be used for fraud and other attacks on banks and their clients

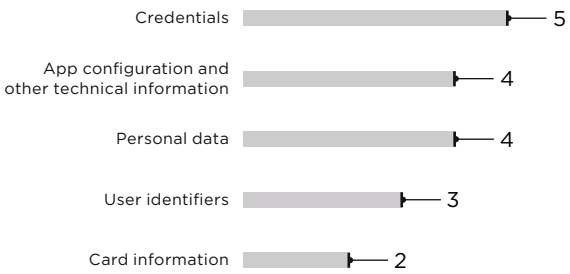

Figure 17. Disclosed information (number of application servers)

What users should know

All mobile banking applications have security flaws. Our research shows that Android apps are more vulnerable than iOS ones. The vulnerabilities that hackers exploit for fraud and theft are usually the result of coding errors. Avoiding such flaws should be top priority for developers. However, many vulnerabilities cannot be exploited without user interaction. Some attacks require physical access to the device.

Rooting (Android) or jailbreaking (iOS) a device, or not setting a PIN code to unlock the phone, gives attackers more leverage to conduct malicious actions.

Tip for users

Do not jailbreak or root your device. This opens up access to the device file system and disables data protection mechanisms. Set a PIN code to unlock your device. This limits attackers' options even if they have physical access to your phone.

Some attacks require user interaction in the form of clicking a link, installing malware, or entering data on a fake web page.

Tip for users

Do not open links sent by strangers via SMS or chat. Never sideload applications from unofficial sources. Download applications only from official stores like Google Play and the App Store. When deciding what to download, pay attention to information on the app developer and the number of downloads.

Vulnerabilities can also reside in the mobile OS itself. But Google and Apple constantly update their software and release security patches. Users should remember that vulnerabilities become public after fixes are released. Hackers can make use of this to attack devices that don't have the latest updates installed.

Tip for users

Always install the latest updates for your OS and mobile applications.

Conclusion

None of the tested mobile banking applications has an acceptable level of security. Banks are not protected from reverse engineering of their mobile apps. Moreover, they give short shrift to source code protection, store sensitive data on mobile devices in cleartext, and make errors allowing hackers to bypass authentication and authorization mechanisms and bruteforce user credentials.

As shown here, mobile banking applications contain flaws that can lead to the following consequences:

- Breaches of sensitive user information, including personal data and card details
- Unauthorized access to the application
- Fraud and theft of funds

Securing data and funds is not just the job of developers; users, too, play a vital role in keeping themselves safe. Most attacks are impossible without user interaction. In 87 percent of cases, user interaction is required for a vulnerability to be exploited. By jailbreaking or rooting, sideloading applications from unofficial sources, visiting suspicious websites, and following dodgy links from SMS and chat messages, users actually help hackers and put their data at risk.

We continue to urge that banks do a better job of emphasizing application security throughout both design and development. Source code is rife with issues, making it vital to revisit development approaches at all stages of the application lifecycle in order to avoid security gaps and ensure strong implementation of SSDL practices. Some vulnerabilities, especially those related to application logic, are impossible to predict. This is why we also recommend thoroughly testing applications and their security mechanisms, with proper attention paid to source code analysis.

ptsecurity.com
info@ptsecurity.com