5 Steps to Securing Neobanks and their Customers

2023

ZIMPERIUM®

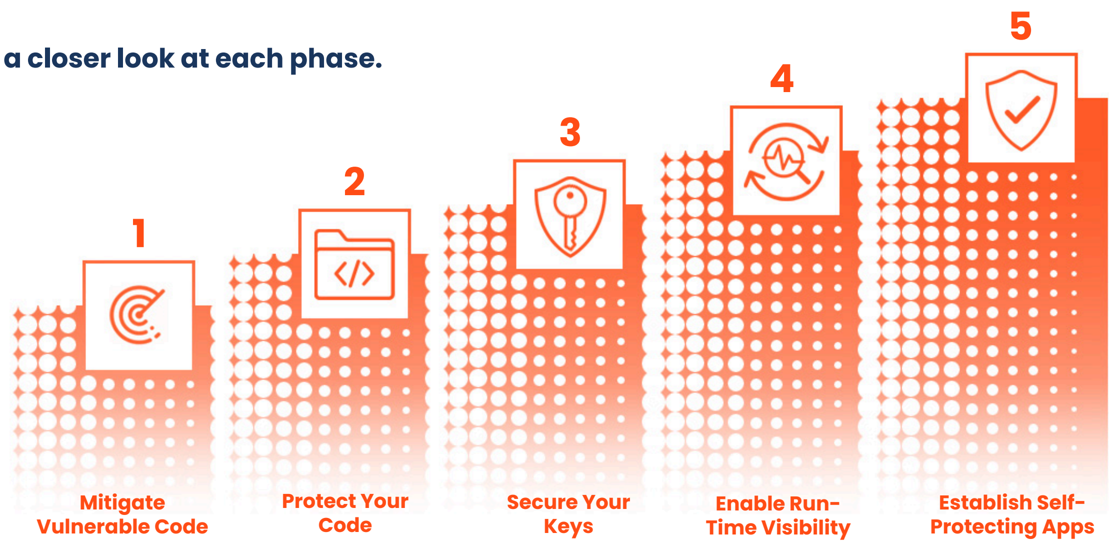
The banking industry has seen significant changes in the past decade with the rise of digital banks, also known as challenger banks or Neobanks. Such digital banks have a common characteristic, which is that they rely on modern technology to offer their services to both businesses and consumers - ***without the need for physical offices.*** Such services can range from checking and saving accounts, money transfers and currency exchange, peer to peer (P2P) and tokenized payments, lending, cryptocurrency, debit and credit cards, and more.

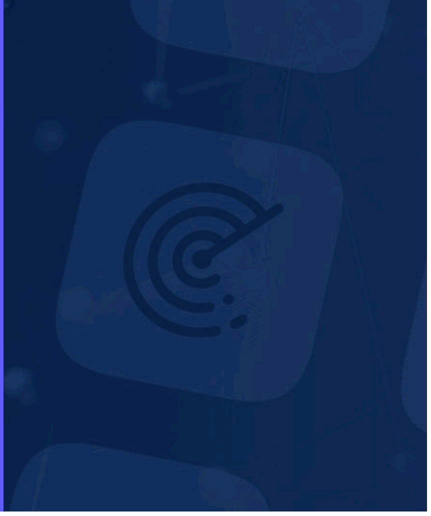
As Neobanks don't have physical offices, they rely on digital 'Know Your Customer' (KYC) processes to validate someone's identity. This is a key difference from traditional banks which often require in-person validation of someone's identity and papers. As a result, Neobanks are likely more vulnerable to different forms of abuse and fraud including (but not limited to) 'New Account Fraud', 'Account Takeover Fraud', 'Loan Application Fraud' and 'Authorized Push Payment Fraud'.

To manage such types of fraud and comply with applicable regulation (e.g. PSD2, AML and other regulatory requirements), these apps require more than only an effective way to validate the end-users identity during enrollment and usage of the app. The Digital KYC and authentication functionality (often 2FA or MFA), as well as the core functionality, needs to be able to withstand sophisticated attackers applying reversing, app tampering, and advanced malware.

To meet these requirements, app developers and providers need to pursue a five-phase approach. By progressing through these steps, development teams can gain the maturity required to most effectively secure their apps and the sensitive data used in their apps.

Let's take a closer look at each phase.





STEP #1

Mitigate Coding Risks

To establish strong defenses, it is vital that development teams start by understanding the risks confronting their apps, including where and how they could be attacked. Too often, however, organizations don't consider security until just before a release. For example, it may only be when code is about to be sent for production release that penetration testing will be done, often uncovering a significant number of vulnerabilities. Unfortunately, not only does this last-minute approach introduce risk, ***it is also typically far more expensive*** and time-consuming to address vulnerabilities the later code gets in the development lifecycle.

How to Succeed

Security requirements must be considered and addressed before, during, and after code is written. By harnessing continuous testing, teams can identify issues and enable fixes throughout the development lifecycle, helping streamline processes while also improving security. In addition, by developing apps securely from the coding stage, teams can reduce risk and avoid the cost, effort, and ***delays associated with addressing vulnerabilities later*** in the software lifecycle. Following are some keys to success:

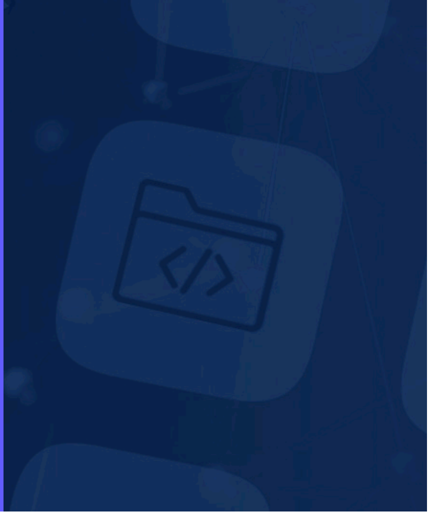
Leverage automated assessments. If teams rely solely on manual penetration testing approaches, it will slow down development and compromise staff efficiency. Further, testing won't be as comprehensive as it needs to be and each pen test can be ***costly.*** Teams need to harness automated capabilities that can integrate into development processes so teams can pinpoint the areas of risk. Policy violations should automatically generate a ticket for developers to fix.

Beware of hybrid app development approaches. Increasingly, teams are using languages or coding frameworks to make hybrid apps, that is, developing one codebase that works on both iOS and Android devices. However, the reality is that ***these types of app frameworks lack many of the security controls available*** in native development environments, so teams should be judicious in adopting these approaches. Similarly, third-party components, whether they're proprietary or open source, can also introduce risks. Quite often, the developers behind these components lack the required expertise and focus on security.

How Zimperium Can Help

Zimperium's zScan can help Neobank developers identify risks in their mobile banking and payment app binaries. With zScan, teams can identify privacy, security, and compliance risks before apps are released to the public. zScan's static and dynamic analysis identifies the specific risk areas an attacker could exploit, including in first-party code, third-party applications, and any third-party components within your application.

zSCAN™



STEP #2

Protect Your Code

Today, it is easy for malicious actors to download an app from an app store, reverse engineer it, find exploitable errors and vulnerabilities, and perform malicious activities, including code injection, piracy, and more. For example, criminals can reconfigure and repackage an app to use in a phishing campaign designed to steal a victim's credentials or ***develop malware specifically targeting your neobanking app.*** Mobile banking app providers simply can't continue to leave their apps vulnerable to this kind of threat.

How to Succeed

Rely on experts. Sometimes, app development teams try to create their own mobile application shielding capabilities rather than using proven commercial solutions. Not only are these efforts very expensive and time-consuming to deliver and maintain, in practice they are also inferior. While internal development teams may have some security experience and internal context, the reality is that they don't have the expertise and resources to ***manage security on their own*** over the lifetime of an app. In addition, security technologies and attacker techniques continue to evolve rapidly. To keep pace and maintain a secure app, it is vital to rely on experts who are solely focused on security.

Employ mobile app obfuscation and shielding. To counter the threats of code compromise, teams must employ mobile app obfuscation and app shielding. Mobile app obfuscation is one of the most critical tools available to developers and security teams. By doing advanced source code obfuscation, teams can make it difficult and time-consuming for potential attackers to determine how the code works. Teams also need to establish robust app shielding capabilities so that, if an attacker bypasses the obfuscation techniques employed, they can't tamper with or bypass business logic to gain access to sensitive data or start to modify the code.

Don't use open source or freeware security tools. These basic tools simply don't provide sufficient resistance to thwart would-be attackers. For example, many open source tools have protection capabilities, but there are easily accessible countermeasures that counteract these safeguards, *such as* ***YouTube videos*** that offer step-by-step instructions and even tooling to help strip the protection capabilities from a 'protected' app.

Get visibility into anti-tampering activities. It's essential for teams to be able to ascertain whether anti-tampering works. Too often, however, mobile banking app developers simply integrate some checks inside the app, and if something malicious were detected, there would be a preset response delivered. For example, a transaction may simply fail or issue a message to the user, indicating the app is down due to potentially suspicious activity. Unfortunately, with these approaches, app providers deliver a poor user experience and lack visibility into whether defenses are working. Therefore, it is vital to monitor anti-tampering activities, ***so teams can be assured that existing mechanisms are working*** or take the steps necessary to respond if not.

How Zimperium Can Help

Zimperium's zShield offers advanced obfuscation and anti-tampering capabilities that enable teams to harden and protect their neobank app code, intellectual property, and private data. Additionally, zShield provides visibility into tampering attempts.

zSHIELD™

STEP #3

Secure Your Keys

Encryption represents a vital line of defense for mobile payment app providers. However, too often, cryptographic keys aren't fully secured, which can weaken or completely negate the benefits of encryption.

How to Succeed

Following are a few critical approaches and considerations for ensuring keys are sufficiently protected:

Don't rely on internally developed key protection. To expedite delivery, teams may simply run a hash over the key material to hide it—but that's *not sufficient.* Alternatively, teams may try to employ their own cryptography algorithms. Through these internally sourced approaches, organizations run the very real possibility of having keys exposed to malware or attackers, leaving critical data and services exposed.

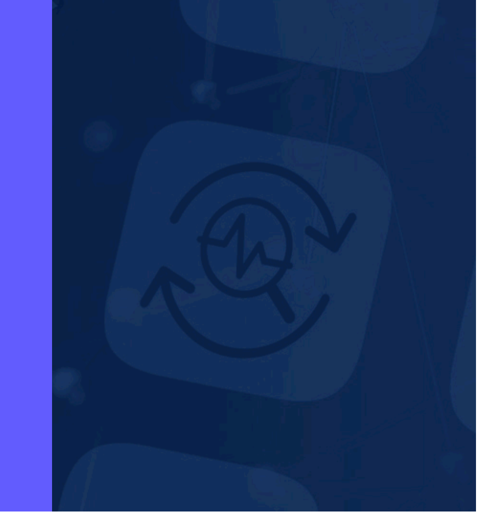
Assure availability of required cryptographic algorithms and protocols, across platforms. Mobile platforms often provide some cryptographic capabilities. These platform capabilities often present several challenges, such as lack of support for specific cryptographic algorithms or operations, infrequent (security) update cycles, and fragmentation of solutions over different mobile platforms. In most cases, third-party app developers can't access the hardware-based Trusted Execution Environments or Secure Enclave, as these are restricted by the smartphone vendor. *Standard available crypto APIs available in the mobile Operating Systems also leave your keys vulnerable and exposed when the device is rooted or jailbroken, or simply stopped receiving security updates.*

Employ white-box cryptography. To establish strong, resilient, and efficient defenses around cryptographic keys, mobile payment and neobank app developers need to employ white-box cryptography. This software-based technology transforms and protects cryptographic algorithms such that keys never appear in the clear, and execution logic is *unexploitable.* Consequently, keys can't be extracted even if the device has been fully compromised.

How Zimperium Can Help

With Zimperium's zKeyBox, development teams establish strong security for their cryptographic keys, without being saddled by the challenges of internally sourced or hardware-based protection approaches. zKeyBox ensures keys are protected, concealed, and never shown in plaintext, *even if an attacker gains complete control* of the execution environment.

zKEYBOX™



STEP #4
Enable Run-Time Visibility

The reality is that there are nearly infinite permutations of devices that may ultimately run a neobanking app. Even the most securely architected and coded app may be exposed by vulnerable operating systems, unsecured networks, and malware.

How to Succeed

To successfully protect apps running on user devices teams need to leverage the following capabilities:

Enable run-time visibility. It is essential to detect and stop attacks when they're happening. This means identifying these attacks on the end user's device at run time using RASP. In effect, teams need to institute trip-wires that flag when an app is being attacked or tampered with.

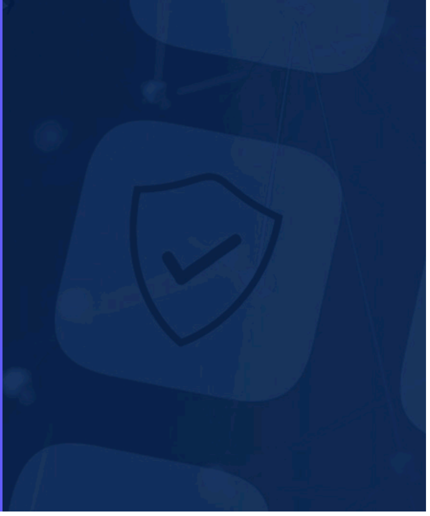
Continuous threat modeling. It is vital that teams establish the visibility needed to detect the nature of the environment that a device is operating in and continuously pull that intelligence into threat modeling. This is essential in determining the optimal course of action and continuing to optimize security over the app lifecycle.

Enable over-the-air (OTA) updates. Mobile banking app developers need a solution that offers the ability to *update their security posture in real-time to keep up with evolving threats and zero-day attacks.* However, without that capability, teams have to recompile the application and make their users constantly reinstall or update their apps, which can be expensive, time-consuming, and inconvenient.

How Zimperium Can Help

Zimperium's zDefend provides continuous monitoring and protection and delivers effective threat modeling capabilities. With zDefend, teams can gain the run-time visibility they need to spot and stop threats before it's too late.

zDEFEND™



STEP #5 Establish Self-Protecting Apps

To reach the ultimate stage of maturity, teams must establish self-protecting apps. At a high level, the goal is to address two potential threats:

- **The theft of data on the mobile device.** In most mobile banking apps today, fraud detection systems are based on monitoring events and transactions to spot suspicious activities. These approaches are largely reactive in nature, raising alerts after a fraudulent transaction has occurred. Through self-protection, teams take a more proactive approach, which means they can avoid the high cost of fraudulent transactions and the resulting clean-up required.
- **The leakage of data.** Often, application logic or constructs can be exploited by malware and other attacks. For example, a cybercriminal may be eavesdropping on a network or employ phishing links. Application self-protection enables these potential risks to be addressed before data is exposed.

How to Succeed

In making their mobile payment apps self-aware, teams need to ensure they avoid the following potential obstacles:

- **Relying solely on anti-malware defenses.** For many enterprise security teams today, protecting apps against malware is a major focus area. Establishing strong anti-malware defenses is a great start, but it isn't enough. The exploitation of mobile banking apps can happen in many other ways.*With anti-malware capabilities alone, teams may, in effect, protect the front gate while leaving the back door open.*
- **Counting on signature-based defenses.** Given all the different malware variants being detected and how quickly they evolve, teams can't afford to rely on signature-based approaches.*These tools offer static mechanisms that simply can't offer the level of security required.* Further, these approaches tend to add significant processing requirements to apps, which can degrade performance and the user experience. Also, signature-based tools often rely on constant cloud-based lookups, which can further degrade performance.

To ensure mobile apps are self-protecting, teams must complete the steps above in order to establish true defense-in-depth capabilities. To be successful, it is essential to stop on-device exploitation in untrusted environments. In addition, teams must understand the sequence of events required to put the app and data in danger. It is also important to understand the threats in the environment so teams can take informed actions on what to do inside apps.

To establish the continuous intelligence today's dynamic environments require, a vast amount of data needs to be tracked and analyzed. To gain timely and actionable intelligence, teams must harness machine learning. *An effective solution must be capable of analyzing an enormous number of events and prioritizing them based on their criticality.* Through machine learning, security teams can reduce false positives and streamline their efforts. As a result, teams can improve detection capabilities and make better decisions about how to respond.

How Zimperium Can Help

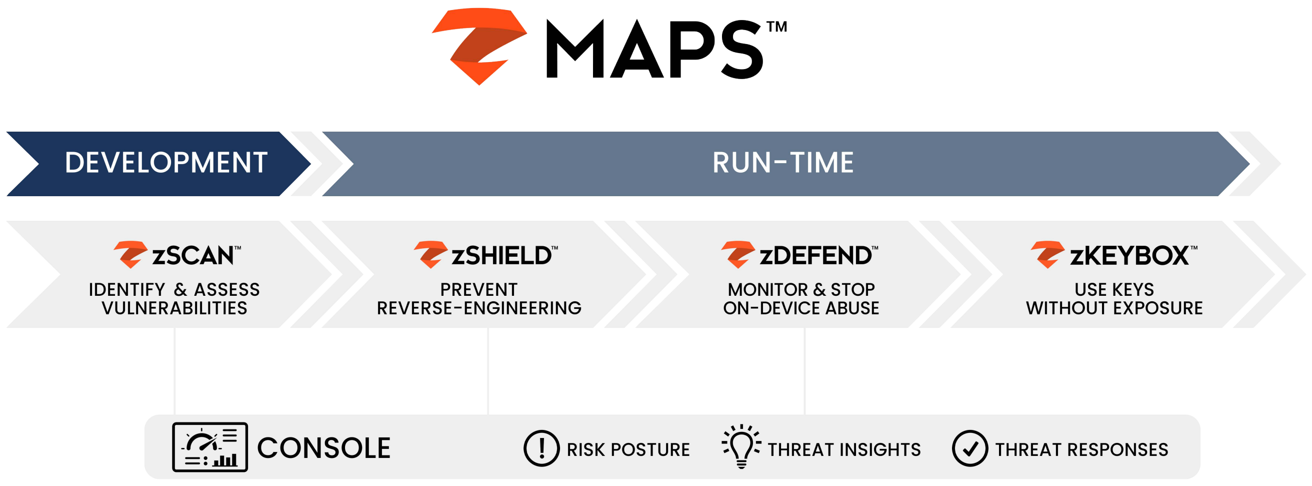
Developers can now detect and mitigate risks more efficiently with Zimperium's zDefend. zDefend is an SDK embedded in the mobile application that enables the host application to attest the device, detect threats, and proactively protect itself. zDefend leverages z9, Zimperium's patented machine-learning-based threat detection engine, to deliver advanced protection to apps.

Conclusion

By taking the five steps described above, teams can begin to holistically and continuously secure mobile banking apps. By mitigating coding risks, safeguarding code and keys, gaining run-time visibility, and establishing self-protection capabilities, teams can realize comprehensive app security. Consequently, mobile banking app providers can effectively secure their apps, customer data, and overall business.

Fortunately, Zimperium offers solutions that can help customers progress through these five steps and do so with maximum speed and efficiency. Zimperium's Mobile Application Protection Suite (MAPS) offers the comprehensive capabilities that teams need to boost the security of their apps. The suite combines comprehensive in-app protection with centralized threat visibility. MAPS includes capabilities for automated application scanning, white box cryptography, anti-tampering and code-hardening capabilities, and advanced, machine-learning based threat detection, attestation and monitoring.

Contact us today to learn more about how Zimperium can effectively and efficiently secure your mobile banking application.


MAPS™
DEVELOPMENT
RUN-TIME
zSCAN™
IDENTIFY & ASSESS VULNERABILITIES
zSHIELD™
PREVENT REVERSE-ENGINEERING
zDEFEND™
MONITOR & STOP ON-DEVICE ABUSE
zKEYBOX™
USE KEYS WITHOUT EXPOSURE
CONSOLE
RISK POSTURE
THREAT INSIGHTS
THREAT RESPONSES

ZIMPERIUM®

Learn more at: zimperium.com
Contact us at: 844.601.6760 | info@zimperium.com

Zimperium, Inc
4055 Valley View, Dallas, TX 75244