How Organizations Can **LOWER** Their Risk

PROTECT
application code

ENCRYPT
critical keys & data

ALERT
organizations to threats

ABOUT ARXAN

Arxan, the global trusted leader of application protection solutions, delivers the confidence to build, deploy, and manage the most innovative and valuable applications

Visit **www.arxan.com** to learn more.

ARXAN
Application Protection

In Plain Sight:

The Vulnerability Epidemic in Financial Mobile Apps

Aitë RESEARCH, COMMISSIONED BY ARXAN

EXECUTIVE SUMMARY

In Plain Sight: The Vulnerability Epidemic in Financial Mobile Apps, commissioned by Arxan and produced by Aite Group, examines the perceived security of financial mobile apps. It highlights the findings from a research campaign that Aite Group conducted over a six-week period to analyze financial institutions' (FIs') mobile apps across every vertical in financial services. The quantity and severity of the vulnerabilities discovered across the mobile apps clearly identify a systemic problem: a widespread absence of application security controls and secure coding, such as technology that implements application shielding, detection, and response capabilities.

Application shielding is a process in which the source code of an app is obfuscated, preventing adversaries from analyzing (aka decompiling) it to find vulnerabilities in the mobile app or repackage it to distribute it with malware. Application shielding also provides other enhanced security, such as application binding, repackaging detection, tamper detection, data-at-rest encryption, and key protection through white-box encryption. App-level threat detection identifies and alerts on exactly how and when apps are attacked at the code level in exercises such as the one Aite Group performed. And threat response can trigger immediate actions, such as shutting down an application, sandboxing a user, revising business logic, and repairing code.

In this research, the mobile applications were decompiled, meaning we reversed the app back to its original source code to assess vulnerabilities. When an app is capable of being decompiled, it allows adversaries to access sensitive information inside the source code—such as application programming interface (API) keys, API secrets, private certificates, and URLs that the app communicates with (which would allow an adversary to then target the APIs of the back-end servers)—recompile it to insert malware for later redistribution, and gain an understanding of how it detects jailbroken/rooted phones so they can circumvent those checks and disable mandatory code signing and sandboxing.

The **AVERAGE TIME TO CRACK** into 30 Mobile Financial Services Apps

“

The **quantity and severity of the vulnerabilities** discovered across the mobile apps **clearly identify a systemic problem: a widespread absence of application security controls and secure coding**, such as technology that implements application shielding, detections and response capabilities.

30 APPS

across every financial services vertical were vulnerable to reverse engineering attacks that expose app code, encryption keys and more:

- Retail Bank
- Credit Card Issuer
- Mobile Payment
- HSA Bank
- Retail Brokerage
- Health Insurer
- Auto Insurer
- Cryptocurrency

FIRMOGRAPHICS

Market Cap

Small to over $10 billion US

Employees

From 72 to 250,000

Geographic

25 HQ in US, 5 HQ in Europe*

*Geographic breakdown of targeted apps' company headquarters (N=30)

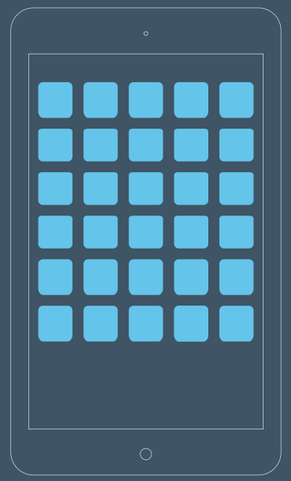
METHODOLOGY

30 Android Apps
Downloaded from Google Play Store*

Publicly available tools
to unpackage and analyze app code:

- APK Extractor
- Apktool
- MobSF
- Burp Suite

Running on an LG G Pad X2



*Methodology could be applied to iOS apps downloaded from the Apple App Store to conduct status code analysis and reveal similar vulnerabilities.

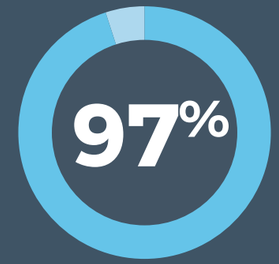
FINDINGS

11 types of vulnerabilities

- Lack of binary protections
- Unintended data leakage
- Weak encryption
- Execution of activites using root
- World readable/writable files and directories
- Insecure random number generation
- Insecure data storage
- Client-side injection
- Implicit trust of all certificates
- Private key exposure
- Exposure of database parameters and SQL queries

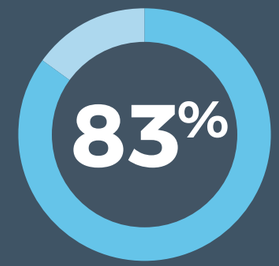
KEY FINDINGS

Lack of Binary Protections
97% of all apps tested lacked binary code protection, making it possible to reverse engineer or decompile the apps exposing source code to analysis and tampering



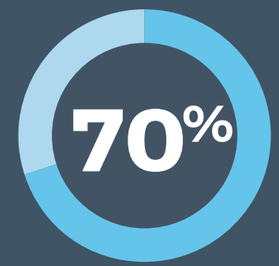
Unintended Data Leakage
90% of the apps tested shared services with other applications on the device, leaving data from the FI's app accessible to any other application on the device

Insecure Data Storage
83% of the apps tested insecurely stored data outside of the app's control, for example, in a device's local file system, external storage, and copied data to the clipboard allowing shared access with other apps; and, exposed a new attack surface via APIs



Weak Encryption
80% of the apps tested implemented weak encryption algorithms or the incorrect implementation of a strong cipher, allowing adversaries to decrypt sensitive data and manipulate or steal it as needed

Insecure Random-Number Generation
70% of the apps use an insecure random-number generator, a security measure that relies on random values to restrict access to a sensitive resource, making the values easily guessed and hackable



TOTAL OF 180

critical vulnerabilities discovered across all apps analyzed

GREATEST NUMBER
of Critical Vulnerabilities

Retail Bank

LOWEST NUMBER
of Critical Vulnerabilities

HSA Bank

FEWEST
Security Controls

HSA Bank

MOST
Security Controls

Cryptocurrency

> Surprisingly, the smaller companies had the most secure development hygiene, while the larger companies produced the most vulnerable apps.

SUMMARY OF FINDINGS

Tables A & B: Vulnerabilities Found Across All Financial Sectors

Source: Aite Group

LEGEND

0% of the apps tested exhibited the vulnerability	25% of the apps tested exhibited the vulnerability	50% of the apps tested exhibited the vulnerability	75% of the apps tested exhibited the vulnerability	100% of the apps tested exhibited the vulnerability

VULNERABILITIES	RETAIL BANK	CREDIT CARD ISSUER	MOBILE PAY	HSA BANK	RETAIL BROKER	HEALTH INSURER	AUTO INSURER	CRYPTO-CURRENCY
Lack of binary protections	100%	75%	100%	100%	75%	100%	100%	75%
Insecure data storage	100%	100%	100%	75%	75%	50%	100%	75%
Unintended data leakage	100%	100%	100%	75%	100%	75%	100%	75%
Client-side injection	50%	75%	75%	50%	50%	25%	50%	25%
Weak Encryption	100%	100%	100%	75%	75%	75%	75%	75%
Implicit trust of all certificates	25%	0%	0%	0%	25%	0%	0%	25%
Execution of activities as root	50%	50%	0%	75%	25%	75%	25%	75%
World readable/ writable files & directories	25%	0%	0%	0%	0%	0%	0%	0%
Private Key Exposure	25%	0%	0%	25%	25%	0%	50%	75%
Exposure of data-base parameters & SQL queries	100%	75%	100%	50%	50%	50%	75%	50%
Insecure random number generation	100%	100%	50%	50%	75%	75%	75%	75%

KEY TAKEAWAYS

The most egregious failure
these apps suffered from was the lack of adequate application security technology—enabling apps to be reverse engineered in minutes—which then allows for host of vulnerabilities, keys and other secrets to be discovered

An alarming amount of sensitive data
surrounding API servers including locations and encryption keys were contained within the apps—placing backend systems and data at a significant risk

Rampant insecure coding
when developing mobile apps—providing a clear roadmap of the app structure and logic

Rare protection of sensitive data
in secured or encrypted space—exposing data to surveillance, and exfiltration

No capability to detect reverse engineering
preventing financial institutions from mitigating attacks before they become widespread

TO LEARN MORE

About the vulnerabilities contained in leading financial institutions mobile apps and how they can be remediated—

DOWNLOAD THE FULL REPORT
at arxan.com/aite