For: Tezos Foundation and Tezsure Inc
Shared on demand. 20 December 2023

COSSACK LABS

Plenty wallet security assessment: executive summary

Overview and Executive summary

The security assessment of the Plenty mobile wallet is the second stage of the Plenty wallet's security improvement process. The first stage lasted from January to May 2023 and focused on security assessments and modifications to the Dartez cryptography library.

The second stage, which began in June 2023, is centred on the Plenty mobile wallet's security design and implementation. Plenty wallet uses the Dartez library, with security enhancements identified in the first step.

The Plenty wallet, a mobile self-custodial cryptocurrency wallet for the Tezos ecosystem, provides users with the ability to manage XTZ tokens and interact with decentralised applications (DApps). The mobile application supports Web3 and social authentication. It is written in the Dart programming language and uses the Flutter framework to be available on many platforms, including iOS and Android.

The wallet app implements the following features: account management, token and NFT transactions, exploring dApps, personal NFT gallery, monitoring news related to the Tezos ecosystem, etc. Some of the functionality is provided by the Dartez library written by the Tezsure team. The application also extensively relies on the external and third-party services: blockchain explorers, analytic services, RPC blockchain nodes, etc.

During the assessment phase, Tezsure team has renamed and rebranded the product from Naan to Plenty wallet.

Goals

The assessment goal is to identify security flaws of the application design and implementation, and provide improvements for security and cryptography aspects. The review is focused on preventing massive exploits and transaction fraud.

Risk statement and security claims

As Plenty wallet doesn't have a public set of security claims and risk statement, we assume that its users expect the same level of security as any digital cryptocurrency wallet:

> *Plenty wallet stores users' accounts securely, does not leak secrets (keys, mnemonics, passwords) and sensitive data, prevents unauthorised access and transactions.*

Security components inside Plenty wallet should be sound against the following risk statement (C.A.S.E.):

> *Financial loss (consequence) due to active and passive adversaries (source) exploiting application security and cryptography flaws (event) in Plenty wallet, resulting in unauthorised usage of secrets, keys and other identifying material (assets) to perform unauthorised transactions.*

The risk statement above is limited to the mobile application itself, so the user's mobile device and mobile OS compromise are out of direct scope.

Scope

In scope for risk statement:

- iOS and Android application behaviour (relevant to the threat modelling),
- secure management of sensitive data,
- authentication and session management,
- RPC communication with Tezos Network,
- API calls between the application and external services related to sensitive data flow,
- cryptography design and implementations around transaction signing,
- transactions correctness (transactions falsification, signing, abuse),
- usage of 3rd party libraries and components (relevant to the threat modelling),
- compliance to the current Apple / Google privacy requirements,
- security-relevant development and deployment process (dependency management, vulnerability management, secure CI, infrastructure deploy).

Updated scope for risk statement:

After initial assessment, application resilience against reverse engineering and tampering was included into the assessment scope.

Non-scope for risk statement:

- The functionality and safety of Tezos mainnet.
- Security and safety of 3rd party services and DApps that Plenty wallet communicates with.

- Security of Plenty wallet as a desktop application.
- It is understood that Plenty wallet security controls might be partially or fully compromised if the mobile device is jailbroken / rooted, or is under Pegasus-like attack.
- It is understood that anyone with access to the users' mobile device has a partial or full control over their wallet.
- 3rd party implementations of cryptographic primitives are considered trusted, some of the implementations were reviewed during the Dartez review stage.

Coverage:

This security review has been based on:

- Initial commit for source code: https://github.com/Tezsure/naan-app/tree/e8b5c0e520b23b8473242f10855298fb81502ce8
- Updated source code after critical fixes: https://github.com/Tezsure/naan-app-audit/commit/7aad6ae77e66e10089baada2586b16af3f04b432
- Design documentation and components description provided by the team.

Verification of fixes has been based on:

- Updated source code after anti-RE fixes: https://github.com/Tezsure/naan-app-audit/commit/7d87516cce5da4c9adf4678f833e5fb66cec82c4
- Latest commit by the end of security assessment: https://github.com/Tezsure/naan-app-audit/commit/42531340f8e7364db661101db818c07c7a843ae8

The review was conducted using the following devices: iPhone 13 (iOS 16.6), iPhone 7 (iOS 14.3), iPhone 7 (iOS 15.5), iPhone 11 Pro (iOS 16.5.1), OnePlus Nord N10 (Android 10), Google Pixel 6a (Android 13), Xiaomi Redmi Note 9 (Android 10), AVDs.

Methodology

Cossack Labs' review has constituted of a number of activities:

- **General risk model clarification and security review**: formulating realistic risk models and threat vectors that affect user safety.
- **Design/architecture review**: proactively seeking design flaws that lead to leakage of sensitive data or manipulating transaction flow.
- **Review of Dartez library integration**: reviewing the source code to verify correct integration of Dartez library and that functionality implemented using it follows security and cryptographic best practices.
- **Research of social login and Web3Auth usage in Plenty wallet**: describing security considerations and providing guidance for improvements of existing social login flow.

- **Review of financial transaction flow and fraud detection**, performing fraudulent transactions. Research and review of transaction logic and process flaws.
- **Application, platform, data security:** ensuring that application level security controls are implemented well, researching platform-specific threats, providing recommendations against supply chain risks.
- **Application security review of surrounding tools:** tests, helper scripts, building process. Review of Apple / Google requirements, privacy forms, security.txt support.

Applicable standards

During the assessment, our work was driven by industry experience and (where applicable to a reasonable extent) the following standards:

Review baseline: OWASP MASVS v1.5 (Mobile application security verification standard), OWASP ASVS v4.0.3 (Application security verification standard), CL MSS (Cossack Labs mobile security score, which is a extended and supplemented version of OWASP MASVS).

Industrial standards and recommendations: NIST SP 800-57 (Recommendation for Key Management), NIST SP 800-53 (Security and Privacy Controls for Federal Information Systems and Organizations), NIST SP 800-37 (Risk Management Framework).

Applicable platform standards: Apple platform security guidelines, Android security best practices, community wisdom for Flutter / Dart security practices.

Triaging issues

Due to the specific risk statement/review goal, issues discovered could not be triaged using a common methodology like CWE or CVSS – the outcomes, loss magnitude in the general context are significantly different compared to a regular security assessment and some of the vulnerability severity scores would be misleading.

Issues are triaged as critical, high, medium, or low based on a performed risk assessment and a formulated trust and risk model representing the chosen risk statement.

Findings summary

The resulting list of security findings and recommendations can be found below:

Findings area	Critical	High	Medium	Low
Design	0	2	2	6
Application security	1	7	8	8
Platform trust	0	3	13	9

Cryptography	0	0	1	2
Supply chain	0	0	1	3
Code quality	0	0	0	5
Anti-RE	0	0	5	0
Overall	**1**	**12**	**30**	**33**

Overall findings: 76 findings (1 critical, 12 high, 30 medium, 33 low).
Basic (L1) security requirements coverage: 47% (the apps satisfied 35 security requirements out of 75).

Findings with all levels of severity have been reported, since many low-priority ones may become stepping stones in multi-step attacks. A number of "low" issues are not related to bugs, but rather to enhancements, improvements, and the building of the missing processes and controls.

This assessment was performed during June-September 2023. It consisted of around 370 person-hours of work, allocated between design review, risk/threat modelling, security assessment of implementation for both platforms, and verification of fixed issues.

Verification of fixed issues

Verification of implemented fixes was performed during September-December 2023.

As of 13 December 2023, the resulting number of left issues is the following:

Findings area	Critical	High	Medium	Low
Design	0	1	1	4
Application security	0	0	1	0
Platform trust	0	0	3	0
Cryptography	0	0	0	0
Supply chain	0	0	0	0
Code quality	0	0	0	1
Anti-RE	0	0	2	0
Overall	**0**	**1**	**7**	**5**

Not fixed findings left: 13 findings (0 critical, 1 high, 7 medium, 5 low).
Basic (L1) security requirements coverage: 89% (the apps satisfied 66 security requirements out of 75).

After Tezsure team has fixed the current portion of the acknowledged issues, and Cossack Labs team has verified the fixes, the current status is the following:

- “Fixed”, or “Partially fixed” — 53 findings,
- “Scheduled, not fixed” or “Acknowledged” — 13 findings,
- “Risk accepted” and “Not an issue” — 10 findings.

Full report

The full report package consists of three separate files.

- The final executive summary that provides a high-level overview of the performed audit and found issues (this document).
- Appendix A provides technical details of each finding, its fix and verification status.
- CL MSS table that shows the resulting security score and includes a list of verified requirements, verification results and links to created issues on the Tezsure task tracking board.

Conclusion

We explained the approach, findings and results of the security assessment of the Plenty wallet. We examined the application's secure storage, authentication flow, transaction logic, user interaction, usage of Android and iOS platform features, and communication with third-party services.

We use Cossack Labs Mobile Security Score checklist (CL MSS) to verify that mobile apps address security edge cases and to measure impact of security work during application development. CL MSS is a custom-tailored set of requirements (appropriate to risk model) that includes relevant requirements from OWASP MASVS v1.5 L1 & L2, ASVS v4.0.3 and MITRE ATT&CK.

CL MSS consists of 157 requirements divided into levels:

- basic security requirements (L1), contains 75 items,
- advanced security requirements (L2), contains 61 items,
- reverse engineering protections (R), contains 21 requirements.

The initial assessment has addressed 100% of L1 and L2 requirements, then the additional short engagement has covered 100% of reverse engineering protections (R).

Our impressions after the initial assessment

Our conclusion is that the security controls implemented by the Tezsure team are insufficient to achieve the security claims and prevent risk statements to a high level. Our research concluded that the Plenty wallet contains a number of critical and high issues, the majority of which is related to the core functionality: system design, transactions security, platform and application security.

We found a number of broken/missing security controls that under unfavourable circumstances could lead to wallets or funds being stolen from app's users, transactions modifications, sensitive data leakage and different types of social engineering attacks that can harm the users.

Already implemented controls

The team implemented many baseline security controls, like application locking with passcode, protecting stored user data by keeping it in system-provided encrypted storage, user confirmation on sensitive actions, including transactions, timer mechanisms on screens with sensitive data, etc.

Dartez integration

Tezsure team did a good job integrating Dartez library into the project. As a result, the number of reported cryptographic issues is minimal, with no critical and high issues.

Source code readability

The source code of the project has good readability and cognitive complexity. Just a few low priority improvements were reported to enhance code quality.

Mobile platform security

Almost 40% of our findings are related platform-specific security controls.

The initial implementation of a local authentication system with a passcode contains issues in business logic and lacks brute-force countermeasures. Due to improper input validation in deeplinks, it is possible to bypass the passcode screen, connect to dApps and to sign smart contract messages without authentication.

Platform-specific data storage also has issues in its implementation. The app uses improper data storage for keeping large amounts of data. The storage library doesn't support hardware-backed encryption.

Developers use Flutter, a cross-platform framework for app development. Its usage brings both advantages and disadvantages. Flutter's timer implementation, for example, has issues with backgrounding on iOS. It can potentially lead to data leakage in business logic where security controls rely on timers, such as restricting mnemonics or balancing exposure in the app's UI.

At the same time, adopting Flutter has prevented the situation in which the same security measures are implemented differently on both platforms. Flutter improves maintainability of security controls and application functionality in general.

Application security

About 35% of discovered issues are related to the application security of the app.

TLS certificate validation is disabled in the application source code making the app's network communication extremely vulnerable to MitM and spoofing attacks.

Passcode value and user's private key/mnemonics are stored in the system provided secure storage in unencrypted form. Even though it is highly unlikely for attackers to retrieve this data on an uncompromised device, no one is safe from the Pegasus-like malware, which could eventually steal private keys, leading to loss of user's cryptocurrency.

Further improvements

As the primary objectives of this engagement were to not only to assess, but to improve the product security, we provided design and implementation recommendations aimed at not only in resolving the immediate issues, but laying the groundwork for the project's long-term stability and maintenance.

Our impressions after verification of fixed issues

Tezsure team has addressed the majority of identified issues, which Cossack Labs team has verified and approved them.

Verification process

Tezsure team response time is fast, and the quality of problem resolution is good. In many cases, developers resolve issues quickly and efficiently, and we have no complaints about the chosen mitigation strategies.

However, we noticed a lack of thoroughness and quality assurance of provided fixes. Some issues were reopened several times because the fix introduced new broken functionality on either the Android or the iOS version of the application. Many tickets were reopened due to significant UI errors introduced with the updates.

We also discovered that security fixes often introduced repeating code in the app where similar features have been implemented multiple times with only small changes. For example, AuthenticationService auth_service.dart contains three methods for similar functionality. Despite increasing application security, this approach makes the app more difficult to maintain and update.

Evaluation of third-party libraries

The third-party libraries chosen for secure storage and biometric authentication make it impossible to configure these features in accordance with best practices.

The project makes use of the FlutterSecureStorage library, allowing for use of system-provided secure storage. However, on both iOS and Android, this library doesn't support hardware encryption and doesn't allow binding biometric authentication to Keychain/KeyStore.

Tezsure team has refused to change the library to utilise hardware encryption for highly sensitive data like mnemonics and wallet private keys.

The lack of suitable Flutter libraries that offer this functionality is one of the contributing factors. One potential candidate can securely handle biometric identification for both iOS and Android platforms, satisfy OWASP MASVS requirements, and offer hardware encryption for Android devices.

Cryptography

Another concern is usage of the cryptographic primitives for symmetric encryption for sensitive data storage. AES-CBC is not a viable choice because it is vulnerable to padding oracle and other types of attacks.

Despite getting step-by-step instructions from our team, developers have not followed our recommendations to improve implementation in accordance with security best practices.

Since Plenty Wallet is an open-source project, the encryption algorithm used by the app is publicly available. With enough time and access to the compromised device, attackers will have minimal difficulty retrieving the user's mnemonics and private key by extracting the passcode hash, encrypted data, and salt from the system-provided secure storage.

Biometric authentication

Biometric authentication is implemented insecurely and can be bypassed by a reverse engineer with beginner skills. It might become a bigger issue, especially if the potential target has an outdated device that could be compromised, resulting in a direct loss of the user's assets. Fortunately, the Tezsure team has scheduled a future fix for this issue.

Resilience against reverse engineering

Reverse engineering protections were implemented on the basic level. Application performs simple root and jailbreak detection.

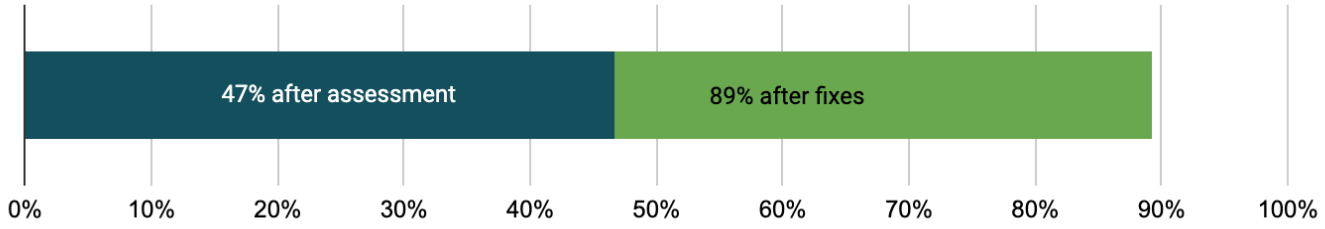
Jailbreak detection on iOS is implemented using a suitable library, but it lacks obfuscation, making it relatively easy for attackers to get around this security control.

Due to the use of the ProGuard, root detection on Android requires more efforts from reverse engineers to bypass this security control and requires the modification of the bypass script on each subsequent build.

Requirements coverage

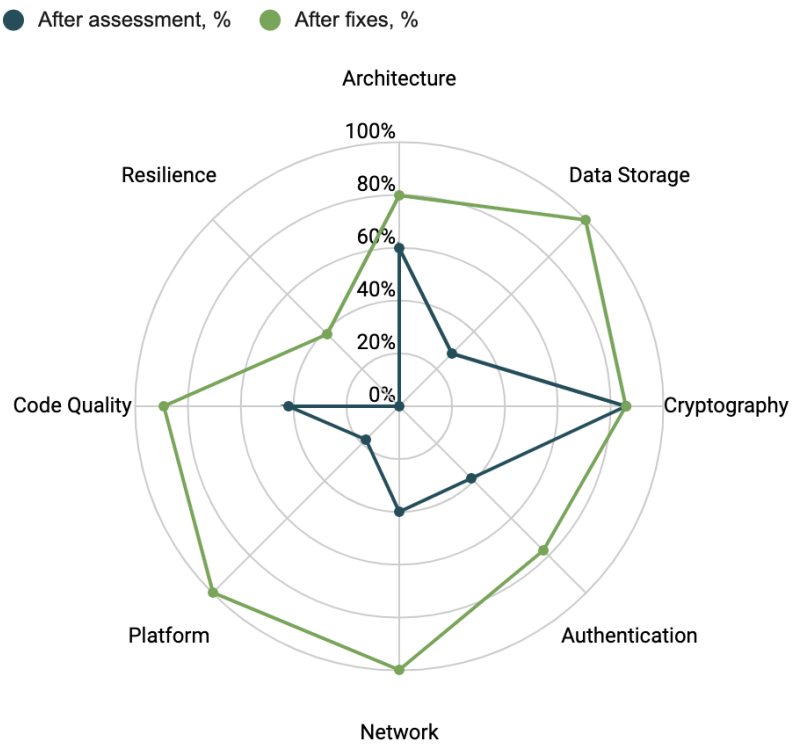
The following diagram represents the progress on fixing the issues after the initial assessment. It is clear that the Tezsure team has covered the majority of requirements by either fixing an issue or accepting the risk.

The number of satisfied basic security requirements grew from 47% to 89% after fixes.



Security requirements covered by Plenty wallet app based on CL MSS: the security score after the assessment compared to the score after fixes verification.

While only basic resilience security controls were implemented, the Tezsure team addressed all issues in the Data Storage, Network and Platform sections of CL MSS. In the “Platform” section, the team was able to go from ~20% of satisfied requirements to 100% in less than 4 months, which we consider an outstanding pace.



Security requirements covered by Plenty wallet app based on CL MSS. Each chapter represents a certain application security topic and has a different number of requirements, calculated as 100%.

We would like to praise Tezsure team's dedication and commitment to improving the security of the Plenty Wallet.

Findings list

We triaged findings into categories depending on the area, type of issue and severity of the outcome.

Types: broken controls (security controls that are implemented but don't satisfy security requirements), missing controls (lack of protections), improvements (security-related suggestions to implement).

Areas:

- Design (D) — architecture or system-level issues.
- Application security (A) — issues found in application-level security controls.
- Platform (P) – platform-specific, mobile-specific issues.
- Crypto (CR) — cryptography related issues.
- Supply chain (S) – CI, 3rd party services, infrastructure around app development.
- Code quality (CQ) — documentation and general code style.

- AntiRE (RE) — detection and protection against application tampering and reverse engineering.

After the verification of the issues, their statuses are the following:

- **Acknowledged** – the issue is acknowledged by the Tezsure team.
- **Fixed / partially fixed** – the issue was confirmed by the Tezsure team, the fix was (partially) implemented in full, and validated by Cossack Labs team.
- **Scheduled, not fixed** – the issue is confirmed by the Tezsure team, planned and scheduled for later, currently not fixed.
- **Not an issue** – the issue was rejected by the Tezsure team due to technological / platform limitations, making it impossible to solve; or the issue was a conscious design decision.
- **Risk accepted** – the issue was Acknowledged by the Tezsure team, but addressing it contradicts current product goals or priorities and might negatively impact the product's effectiveness. The decision was conscious and the security risks were accepted.

ID	Finding	Severity	Type	Area	Status
D-001	Review and comply with Apple In-App Purchase policy	High	improve	design	Scheduled, not fixed
D-002	Missing in-app autolock functionality allows to bypass authentication	High	missing	design	Partially fixed
D-003	Social login via Web3Auth design issues and considerations	Medium	improve	design	Scheduled, not fixed
D-004	Store sensitive data fields in Keychain/Keystore, and space-consuming data in internal database	Medium	improve	design	Fixed
D-005	App does not ask users' consent when processing personal data	Low	missing	design	Scheduled, not fixed
D-006	Multiple issues with wallet mnemonics and private key display	Low	broken	design	Scheduled, not fixed
D-007	Consider new blockchain-based content policy in Google Play	Low	improve	design	Scheduled, not fixed
D-008	Cryptography export regulations:	Low	improve	design	Scheduled, not

ID	Finding	Severity	Type	Area	Status
	filing an annual self-classification BIS report				fixed
D-009	Add security.txt to the mobile project	Low	improve	design	Fixed
D-010	Add application force update feature	Low	improve	design	Fixed
A-001	Don't ignore TLS certificate validation	Critical	broken	appsec	Fixed
A-002	Broken logic of passcode attempts counter	High	broken	appsec	Fixed
A-003	Android: Do not allow cleartext HTTP traffic	High	broken	appsec	Fixed
A-004	Implement Valid Amount Rounding	High	broken	appsec	Not an issue
A-005	Validate Transaction Amount Input	High	broken	appsec	Fixed
A-006	Set more strict access policy for iOS Keychain	High	broken	appsec	Fixed
A-007	User Can Access Secret Phrase Without Password Entrance	High	broken	appsec	Fixed
A-008	Wallet Reset Without Password Entrance	High	broken	appsec	Fixed
A-009	Outdated Server Version	Medium	broken	appsec	Risk accepted
A-010	Missing Input Validation on Wallet Name	Medium	missing	appsec	Fixed
A-011	Don't use floating point numbers for	Medium	broken	appsec	Scheduled, not

ID	Finding	Severity	Type	Area	Status
	handling cryptocurrency				fixed
A-012	Consider using Web3 Auth MFA	Medium	improve	appsec	Fixed
A-013	Old Passcode Value Reuse	Medium	broken	appsec	Fixed
A-014	User can Skip the Password Creation Step	Medium	broken	appsec	Fixed
A-015	Lack of Brute-Force Countermeasures for Application Passcode	Medium	missing	appsec	Fixed
A-016	Educate Users on importance of Secret Recovery Phrase	Medium	improve	appsec	Fixed
A-017	Internal details disclosure in error messages	Low	broken	appsec	Fixed
A-018	Implement Proper Error Handling for Buy Functionality	Low	broken	appsec	Risk accepted
A-019	No TLS certificate pinning	Low	missing	appsec	Risk accepted
A-020	Application Doesn't Handle Transaction Errors	Low	missing	appsec	Fixed
A-021	Server Version Disclosure	Low	broken	appsec	Fixed
A-022	Check HTTP response codes	Low	missing	appsec	Fixed
A-023	Add the Option to Reset Passcode on Login Screen	Low	improve	appsec	Fixed
A-024	Weak Passcode Creation is Allowed	Low	broken	appsec	Fixed
P-001	Application allows to sync wallets	High	broken	platform	Partially fixed

ID	Finding	Severity	Type	Area	Status
	with dApps without authentication				
P-002	Lack of data validation on opening deeplinks	High	improve	platform	Fixed
P-003	Android: Do not allow backups on application level	High	broken	platform	Fixed
P-004	Submit Data Privacy Form	Medium	missing	platform	Scheduled, not fixed
P-005	iOS: Mnemonics timer does not work correctly if app state is changed	Medium	broken	platform	Partially fixed
P-006	Create allow-list for WebView URLs and validate them	Medium	improve	platform	Not an issue
P-007	Clear storage/cache in WebViews	Medium	improve	platform	Fixed
P-008	Allow only https protocol handler in WebView	Medium	improve	platform	Fixed
P-009	iOS: Prevent usage of custom third-party keyboards	Medium	improve	platform	Fixed
P-010	Implement biometry invalidation	Medium	broken	platform	Scheduled, not fixed
P-011	Biometrics is not event-bound	Medium	improve	platform	Scheduled, not fixed
P-012	Hide sensitive data when app moves to the background	Medium	missing	platform	Fixed
P-013	iOS: Clear Keychain on app reinstall	Medium	missing	platform	Fixed

ID	Finding	Severity	Type	Area	Status
P-014	iOS: Review and remove unused permissions	Medium	improve	platform	Fixed
P-015	Drop support of old Android versions: Prevent Overlay attacks	Medium	improve	platform	Fixed
P-016	Android: Review and remove unused permissions	Medium	improve	platform	Fixed
P-017	OneSignal library has remote debugging enabled for the WebView	Low	improve	platform	Fixed
P-018	Android: Accessibility feature abuse prevention	Low	improve	platform	Risk accepted
P-019	Disable screenshots on screens with sensitive data	Low	missing	platform	Fixed
P-020	Use secure alternatives to WebViews if possible	Low	improve	platform	Partially fixed
P-021	FlutterSecureStorage should use hardware encryption	Low	improve	platform	Risk accepted
P-022	Keyboard cache is enabled for sensitive text fields	Low	improve	platform	Partially fixed
P-023	iOS: Prevent buffer overflows, use stack canary	Low	improve	platform	Fixed
P-024	Android: enable minification and resource shrinking	Low	improve	platform	Fixed
P-025	App signing key & certificate protection	Low	improve	platform	Risk accepted
CR-001	Encrypt in-app static secrets	Medium	improve	cryptography	Fixed

ID	Finding	Severity	Type	Area	Status
CR-002	App stores sensitive data in plaintext	Low	improve	cryptography	Partially fixed
CR-003	Hash passcode instead of storing it in plaintext	Low	broken	cryptography	Fixed
S-001	QR code scanning package depends on unmaintained libraries	Medium	broken	supply	Fixed
S-002	Avoid communication with endpoints associated with counties under sanctions	Low	improve	supply	Risk accepted
S-003	Getx dependency is potentially harmful	Low	broken	supply	Risk accepted
S-004	Setup dependency and vulnerability management process	Low	broken	supply	Fixed
CQ-001	Android: Production APK contains unused files	Low	improve	code	Scheduled, not fixed
CQ-002	Application contains dead/unused code	Low	improve	code	Fixed
CQ-003	App requests biometric authentication too many times	Low	broken	code	Fixed
CQ-004	Strip debug symbols for release builds	Low	improve	code	Fixed
CQ-005	Don't use print statements in release	Low	broken	code	Fixed
RE-001	Validate iOS device with App Attest	Medium	missing	antiRE	Scheduled, not fixed
RE-002	Validate Android device with Play Integrity	Medium	missing	antiRE	Scheduled, not fixed

ID	Finding	Severity	Type	Area	Status
RE-003	Lack of reverse-engineering protections for Android	Medium	missing	antiRE	Partially fixed
RE-004	Lack of reverse-engineering protections for iOS	Medium	missing	antiRE	Partially fixed
RE-005	Add Flutter and Android source code obfuscation	Medium	missing	antiRE	Partially fixed

About Cossack Labs

Cossack Labs is a provider of data security tools (cryptographic and data security frameworks), bespoke solutions and consulting services, with a focus on sensitive data protection in modern systems. Cossack Labs' experts participating in this audit, have decades of hands-on practical experience, appropriate formal education and academic degrees in cryptography, software engineering, data security and general information security. Cossack Labs' security engineers are acknowledged contributors to popular industry standards (OWASP MASVS/MASTG) and hold appropriate certifications (CISSP, SSCP).

Due to the nature of our skillset, our review aims not only to detect potential weaknesses but also to provide clear, actionable advice for developers to rapidly improve security in their applications, as communicated by thinking-alike engineers.

Cossack Labs can be contacted at: cossacklabs.com / info@cossacklabs.com

0.1	27 September 2023	Cossack Labs team	Initial version of the executive summary and appendix with technical issues.
0.2	5 October 2023	Cossack Labs team	Added issues about reverse engineering protections.
0.3	13 December 2023	Cossack Labs team	Final version of executive summary, appendix with technical issues and CL MSS checklist.
0.4	20 December 2023	Cossack Labs team	Change the phrasing so that Plenty is used as the brand name everywhere.

