Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Exodus iOS Mobile App 03.2019

Cure53, Dr.-Ing. M. Heiderich, BSc. C. Kean, BSc. T.-C. Hong, Prof. N. Kobeissi, Dipl.-Ing. A. Aranguren

Index

Introduction

> *“We poured our hearts into every detail, from pixel-perfect icons to subtle sounds, creating a blockchain asset experience that works for everyone. Exodus makes it fun and easy to learn and use blockchain assets. No technical talk. No messy wallets. No confusing steps. Exodus is designed for people who have never used an exchange. Ready to exchange Bitcoin for Ethereum? Exodus hides the complex details; in seconds, assets are exchanged behind the scenes.”*
>
> From https://www.exodus.io/

This report documents the findings of a security assessment targeting the Exodus mobile applications. Carried out by Cure53 in March 2019, this project entailed both a penetration test and a dedicated audit, additionally placing a strong focus on the iOS mobile apps in the Exodus compound. Only three issues with very limited exploitation potential have been discovered during this assignment.

As for the resources, the assessment was carried out by five members of the Cure53 team. A budget allocated to the project stood at a total of fifteen person-days of work,

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

including preparations, main testing, communications and this report. Under a white-box methodological premise, Cure53 was granted access to the relevant sources of the Exodus mobile applications. In addition, the testers were provided with details about the main points of interest and a demarcation of the scope from a perspective of the Exodus maintainers.

In a very clear setup, top priority was assigned to the Microsoft OTA via CodePush, secure seed and secret storage, as well as secure UI and PIN handling. This tripartite design of main areas in focus was accompanied by general code auditing, as well as investigations honing in on dependencies and rollout. The shift to the less crucial areas occurred when the three main realms were deemed as covered sufficiently. Based on the above outline, Cure53 divided the work into four Work Packages and the corresponding tasks were delegated to team members with the most suitable expertise.

Expanding on this WP structure and the white-box method, the Cure53 team also received currency from the Exodus team as part of the test. This enabled sending actual transactions and the funds will be returned to Exodus in a secure manner in due course. The currency was returned after the test finished.

The project progressed in a productive and timely fashion, benefitting from a clear scope, good preparations and ongoing assistance from the Exodus team. The Cure53 testers communicated with the in-house team throughout the project, using a shared Slack channel explicitly set up for this assessment by the Exodus team.

As noted, the findings are few and far between despite a very thorough coverage and considerable efforts of the testing team seeking for a compromise. While none of the findings carried noteworthy risks, one was considered to be an actual vulnerability with an impact level set to “*Low”.* The remaining two discoveries were classified as general weaknesses and ascribed with “*Informational”*-only value. Moreover, the aforementioned vulnerability was later flagged as a false alert stemming from a minor scope confusion at the very beginning of the test. In that sense, the ultimate outcome of this assessment is exceptional, with basically no security-relevant findings to report.

In the following sections, the report will first present the scope and then moves on to a comprehensive discussion of methodologies and testing strategies. Next, the findings are briefly discussed on a case-by-case basis, and then followed by some concluding words on the general security posture of the Exodus mobile applications tested by Cure53 during this March 2019 project.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Exodus Mobile Apps for iOS**
  - Cure53 was given access to the sources
  - Cure53 was given some crypto currency from the Exodus team to test with
  - Information on scope and focal areas was shared by the Exodus team

Test Methodology

This section describes the methodology used during this source code audit and penetration tests. The test was divided into two phases. Each phase had goals that were closely linked to the areas in scope.

The initial phase (Phase 1) mostly comprised manual source code reviews, in particular in terms of the API endpoints, input handlers and parsers. The review carried out during Phase 1 aimed at spotting insecure code constructs. These were marked whenever a potential capacity for leading to buffer corruption, information leakages and other similar flaws has been identified. In addition, Cure53 dedicated due attention to vital areas specified by the Exodus team prior to the commencement of Phase 1.

The secondary phase (Phase 2) of the test was dedicated to classic penetration testing. At this stage, it was verified whether the security promises made by Exodus in fact held against real-life attack situations and malicious adversaries. This included watching out for disclosure of sensitive information and anything that might mean that an attacker obtains illegitimate access to the user's wallet features.

Phase 1. Manual Code Auditing

The following list presents the steps undertaken during the first part of the test, which entailed the manual code audit of the sources pertinent to the Exodus software in scope. This is to underline that, in spite of the almost nonexistent findings, substantial thoroughness was achieved and considerable efforts have gone into this test. Note that a given realm of work yielded no results unless otherwise indicated with a specific link to a finding.

- The implementation and bootstrapping of Microsoft CodePush was checked to verify its correctness across the stack.
- Secure seed and key storage have been verified by looking into *React Native*'s bindings into the local keychain API and the configuration of those bindings.
- A general audit was carried out in regard to modules handling sensitive operations like seed generation, seed storage, transfers, etc.
- API communications were verified to occur over an encrypted transport layer.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

- The underlying dependency libraries were analyzed and audited. Particularly the employed versions of the external libraries were examined for being free of security issues. EXO-01-001 was filed but later excluded.
- Internal dependency libraries were audited.
- Sensitive operations were reviewed for logic flaws, such as the possibility of avoiding a PIN confirmation prompt.
- Potential for abuse around the process of receiving transactions was found void due to the fact that the blockchain is simply consulted.

In general, a great number of good design decisions were found to be in place during the test. The application effectively reduces the attack surface as much as possible.

Phase 2. Code-Assisted Penetration Testing

A list of items below elaborates on the steps undertaken during Phase 2 of the test. This encompassed code-assisted penetration testing against the Exodus mobile wallet in scope. Given that the manual source code audit did not yield any findings, the second approach was added as means to maximize the test coverage. As for specific tasks taken on to enrich this Phase, these can be found listed and discussed in the ensuing list.

- It was verified that the application does not employ any URL handlers. This was done because, in some cases, URL handler might lead to entry points for attacks like storage pollution and Denial-of-Service.
- The local storage of the Exodus iOS branch was examined for potential leakage of sensitive information such as PINs, private keys or seed phrases.
- The compiled *Info.plist* file was checked to further ensure the configurations for Microsoft CodePush are proper and correct.
- PIN protection was checked to ensure it meets the proper length requirement. Initial test revealed a one-digit PIN code could be used and follow-up tests revealed non-digit PIN code could cause the PIN not being saved. The issues were addressed in EXO-01-002
- UI security practices were examined. It was checked whether the app would be locked when pushed to the background.
- Backup seeds handling was examined. It was determined that the secret words should be excluded from being screen-captured, as the photo could be likely uploaded to the Internet via iCloud. This is discussed in EXO-01-003
- Upon close inspection of the mobile app traffic, it was found that only encrypted communications were being used. A number of MitM attempts were made to ensure that the application will not accept invalid TLS certificates, for example

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

using a self-signed certificate or a CA-signed certificate for a wrong hostname. The app was found to validate TLS certificates correctly.

- Further, the currency exchange requests were examined in regard to requesting and receiving a currency exchange. It was demonstrated that an encrypted connection is utilized and the parameters cannot be manipulated in a way compromising the transaction. Moreover, the wallet is reading the respective amount of the wallet's address correctly from the blockchain.
- Potential for data leakage via backups and file storage was reviewed and also yielded no results.
- In addition to all of the above, all of the explored abuse and compromise scenarios were found to be significantly mitigated due to the fact that the application prompted the user for the PIN prior to making any sensitive operation, such as sending funds to a third-party.

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in a chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *EXO-01-001*) for the purpose of facilitating any future follow-up correspondence.

EXO-01-001 Mobile: react-native-keychain lacks security levels on Android *(Low)*

It was found that the version of *react-native-keychain* employed by the Exodus application (i.e. 3.0.0 version) does not correctly implement security levels when used to create targets for the Android devices. This has been fixed in the 3.1.0 version of *react-native-keychain*[1].

We recommend updating to the latest version of the *react-native-keychain* in order to ensure that potential future deployments of Exodus on Android devices do not suffer from an incomplete ability of setting security parameters for key storage.

Note: This ticket was later identified as a false alert. It has been deemed free from having any security impact on the application. The cause was a slight scope confusion at the beginning of the test.

[1] https://github.com/oblador/react-native-keychain/compare/v3.1.0...master

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

EXO-01-002 iOS: Passcode & lock observations *(Info)*

During testing of the Exodus iOS branch, it was observed that the PIN code had no minimum character requirement and the app did not lock when pushed to the background.

However, the discussions with the development team revealed the following measures that are already planned as means to tackle those issues:

- The passcode will have a requirement of consisting of six digits.
- The background will use Appstate[2] to lock the app when pushed into the background.

Both issues were addressed while testing continued.

Additionally, a UI bug was found when six dots has been used as the passcode. Although the app saves the passcode, it is cleared when the app is reopened. This issue has also been addressed later in the testing.

Note: *Cure53 reviewed the PRs that were created to address this issue. This flaw has been successfully eliminated.*

EXO-01-003 iOS: Observations about screenshot protections *(Info)*

Exodus currently prevents users from taking a screenshot for a paper backup (seeds). This is done by requiring users to press and hold a "*reveal*" button to disclose secret words. This can be abused due to a quirk on iOS, since taking a screenshot there cancels all touches on the screen. It was also found that iOS waits for roughly one second between users pressing the screenshot buttons and actually taking a screenshot.

As a result, users can still take a screenshot of their secret words by pressing the *screenshot* buttons. From then, the *reveal* button would need to be quickly pressed and held. Albeit the chance is slight, it is possible that users accidentally trigger this

[2] https://facebook.github.io/react-native/docs/appstate

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

sequence of actions, especially given the placement of the *screenshot* buttons on the new iPhone models.

Although users have various means of circumventing this measure (e.g. video recording from another device), executing them directly on the phone could cause leakage onto the Internet (i.e. iCloud sync). It is still recommended to fix this problem by, for example, introducing a delay between pressing the "*reveal*" button and showing the secret words.

Conclusions

From both a marginal number of issues and their very limited severities, it can be derived that this Cure53 assessment concludes with a positive verdict about the state of security at the Exodus mobile applications and their corresponding elements. After spending fifteen days on investigating the scope in March 2019, four members of the Cure53 team can attest that every precaution has been taken during the development and subsequent maintenance of the Exodus compound.

To reiterate, the main priorities for this audit were the verification of the correct deployment of the Microsoft CodePush for over-the-air updates, asserting that secure seed and secret storage use on-device keychains in a safe and sound manner, as well as checking if the implementations hold - from a security standpoint - for secure UI and PIN handling. Over the course of the project, Cure53 was able to assess and confirm the correctness of these three aspects. Despite thorough coverage, Cure53 found attacks unfeasible.

Moreover, the coverage extended to other, more general aspects of the codebase shared by Exodus. The code base was found to follow best practices of the *React Native* framework, as well as code conventions. This makes the application easy to audit and, more importantly, ensures that it remains easy for the Exodus team to maintain in the future.

To give more details, session secret management and storage of Exodus uses the *react-native-keychain* library. While it was found that the outdated version of the library employed by Exodus left it vulnerable to a potential weakness in Android deployments (see EXO-01-001), this may not be immediately relevant due to the current application target being restricted to iOS only. In the end, this issue was only filed because of an aforementioned scope confusion, making it a false alert only kept in this report for the sake of completeness.

Few minor, operational security issues were observed in the Exodus application's smartphone interface. First up, EXO-01-002 discusses marginal behavioral bugs that

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

can be triggered by certain PIN entries. Secondly, EXO-01-003 demonstrates how the screenshot prevention can also be bypassed by users, although it is debatable as to which degree this issue may (or should) be addressed.

Similarly, Exodus's API communications make use of encrypted communications via TLS and validate certificates correctly. Certificate pinning could be considered to further protect users against Man-in-the-Middle adversaries capable of obtaining falsely issued certificates. Finally, the Exodus application was found to maintain a sensible security policy in regard to the generation of user-wallets and in terms of the users' ownership of private keys linked to their wallets. For example, private keys are neither stored on the Exodus servers, nor made to leave the device in any way.

The results of this Cure53 assessment underline that Exodus is a very strong candidate for a highly-secure cryptocurrency wallet. The team tackling this project in March 2019 is generally not likely to issue praise but the results of this assessment are outstanding and came as a surprise, especially given the scope objects' complexity and purpose. In other words, having successfully defenses against such a wide array of problems is not common in the realm of cryptocurrency wallets. To sum up, Cure53 concludes that the Exodus application and API are solidly engineered and more than fit-for-deployment as far as security is concerned.

Cure53 would like to thank JP, Ain and Faris from the Exodus team for their excellent project coordination, support and assistance, both before and during this assignment.