verichains

SECURITY AUDIT OF

RONIN MOBILE WALLET

Public Report

Mar 18, 2022

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Mar 18, 2022. We would like to thank Sky Mavis for trusting Verichains Lab in auditing the mobile wallet. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Ronin Mobile Wallet. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the application, along with some recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY **5**

1.1. About Sky Mavis **5**

1.2. About Ronin Mobile Wallet **5**

1.3. Audit scope **5**

1.4. Audit methodology **6**

1.5. Disclaimer **7**

2. AUDIT RESULT **8**

2.1. Overview **8**

2.2. Findings **8**

2.3. Issues **9**

2.3.1. Failed attempt locked time is bypassable MEDIUM 9

2.3.2. ActionQueue may cause application to crash LOW 9

2.4. Possible enhancements **11**

2.4.1. Remove usage of console.log in production INFORMATIVE 11

2.4.2. Change set* to increase* INFORMATIVE 11

2.4.3. Use global instead of window in React Native environment NONE 11

2.4.4. Rooting 12

3. VERSION HISTORY **13**

1. MANAGEMENT SUMMARY

1.1. About Sky Mavis

Sky Mavis is a blockchain gaming studio that built the successful NFT-based online game Axie Infinity. The team has a long history in gaming, they lauched Axie Infinity in 2018 with the idea of a blockchain-based play-to-earn model could create more aligned incentives between game creators and game players in long-term.

The company built Ronin Sidechain to solve network congestion problem for Axie, which provides:

- Fast & seamless transactions with almost instant confirmation.
- Drastically reduced gas fees. In addition, rather than paying Ethereum miners - the gas fees could be retained by the community and used for things like tournaments & bounties.
- The ability to withdraw Axie assets back to Ethereum Mainnet (eventually).
- Simplified on-boarding for new users, through a customized wallet solution.
- A block explorer for transparency and data accessibility.

1.2. About Ronin Mobile Wallet

Ronin Mobile Wallet is the official mobile wallet for Sky Mavis's Ronin sidechain.

This extension allows users to play Axie Infinity and other decentralized applications running on Ronin, an Ethereum sidechain built specifically for Blockchain games. Users can use **Ronin Wallet** to:

- Manage digital identity, experience 100% true ownership of their assets.
- Send transactions without paying expensive gas fees.

1.3. Audit scope

In this particular project, a timebox approach was used to define the consulting effort. This means that **Verichains Lab** allotted a prearranged amount of time to identify and document vulnerabilities. Because of this, there is no guarantee that the project has discovered all possible vulnerabilities and risks.

Furthermore, the security check is only an immediate evaluation of the situation at the time the check was performed. An evaluation of future security levels or possible future risks or vulnerabilities may not be derived from it.

1.4. Audit methodology

Verichains Lab's audit team mainly used the **Open Web Application Security Project (OWASP) Mobile Security Testing Guide (MTSG)**. The **MSTG** is a comprehensive manual for mobile app security development, testing and reverse engineering. It describes technical processes for verifying the controls listed in the **OWASP Mobile Application Verification Standard (MASVS)**. During the audit process, the audit team also used several tools for viewing, finding and verifying security issues of the app, such as following:

#	Name	Version
1	Mobile Security Framework (MobSF)	v3.3 beta
2	Frida tools	14.2.2
3	Android Studio	4.0
4	Visual Studio Code	1.15.1
5	Android Debug Bridge (adb)	1.0.41

Table 1. Tools used for audit

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the application functioning; creates a critical risk to the application; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the application with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the application with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 2. Severity levels

1.5. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure application. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The initial review was conducted in September 2021 and a total effort of 2 weeks was dedicated to identifying and documenting security issues in the code base of the Ronin Mobile Wallet.

The following files were made available in the course of the review:

FILE	SHA1 SUM
mobile-wallet-main.zip	6FB7E05D7F323DAF3868C96F8A1AE4A2A5A0D6C7

2.2. Findings

During the audit process, the audit team did not discover any real security vulnerability issues in the Ronin Mobile Wallet. Only some minor issues, enhancement in coding were found and listed in the following section.

#	Issue	Severity
1	Failed attempt locked time is bypassable	MEDIUM
2	ActionQueue may cause application to crash	LOW

Audit team also suggested some possible enhancements.

#	Issue	Severity
1	Remove usage of console.log in production	INFORMATIVE
2	Change set* to increase*	INFORMATIVE
3	Use global instead of window in React Native environment	NONE

Sky Mavis fixed the code, according to Verichains's private report.

2.3. Issues

2.3.1. Failed attempt locked time is bypassable MEDIUM

Ronin Mobile Wallet implements a lock mechanism which lock passcode attempt when someone enters wrong passcode too many times (password guessing/bruteforce attack), and the locked time is defined as in the snippet bellow:

```
35  state.appLockedAt = new Date().getTime()
```

Snippet 1. app/services/store/models/account.ts

The problem is new Date().getTime() of React Native is untrusted, attackers can modify system datetime to a future timestamp (after release time) to bypass the time restriction and continue to try another passcode. This approach does not require rooting and can be automated.

RECOMMENDATION

Instead of new Date().getTime() based time measuring, it is recommended to implement secure date time measuring using native module combination of time-synchronization from trusted source and local real-time clocks APIs like SystemClock.elapsedRealtime and SystemClock.elapsedRealtimeNanos on Android, mach_absolute_time on iOS. These return the elapsed time since the system was booted, including time when the device goes to deep sleep. This clock is guaranteed to be monotonic and continues to tick even when the CPU is in power saving mode, so is the recommended basis for general purpose interval timing.

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

2.3.2. ActionQueue may cause application to crash LOW

The ActionQueue class in *ActionQueue.ts* does not have cap, so when dapps request signing too many times, it can cause a DOS attack and make the wallet crash, or just stop the user from interactive with other dapps.

```
37  add(actionData: ActionData) {
38      const id = this.nextId
39      this.actions.push({ ...actionData, id, createdAt: Date.now() })
40      this.nextId++
41
42      this.emit(Event.ActionAdded, { id, actionData })
43      return id
44    }
```

Snippet 2. app/services/engine/ActionQueue.ts

RECOMMENDATION

Add a key-based cap for the class:

```
const MAX_CAP = 9;
...
  requestCnt = {};
  requestId2Source = {};
  add(actionData: ActionData, source: string) {
    if (requestCnt[source] >= MAX_CAP) throw new RateLimitError();
    requestCnt[source] = (requestCnt[source] || 0) + 1;
    ...
    requestId2Source[id] = source;
    ...
  }
  public rejectAction(id: number) {
    const source = requestCnt[source];
    if(typeof source !== 'undefined') {
      requestCnt[source]--;
      delete requestId2Source[id];
    }
    ...
  }

  public approveAction(id: number) {
    const source = requestCnt[source];
    if(typeof source !== 'undefined') {
      requestCnt[source]--;
      delete requestId2Source[id];
    }
    ...
  }
```

Snippet 3. Possible fix for ActionQueue class

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

2.4. Possible enhancements

2.4.1. Remove usage of console.log in production INFORMATIVE

Debug function console.log is used in various places:

- *services/engine/KeychainWalletSubprovider.ts:45*
- *services/wallet-connect/WalletConnectManager.ts:146*
- *screens/auth/SyncExtension.tsx:115*

That function is intended for debugging, it is not recommended in production.

RECOMMENDATION

Consider add a general way to handle/remove log messages in production environment.

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

2.4.2. Change set* to increase* INFORMATIVE

Actions that update the failed attempts should be coded in relative-updating way instead of absolute to prevent race-conditions-alike bugs, for example, there can be two places captured an old counter, and then both increasing it, results in an increasement by 1 instead of 2.

```
56  accountActions.setPasscodeFailedAttempt(passcodeFailedAttempt + 1)
```

Snippet 4. app/screens/auth/UnlockWallet.tsx

RECOMMENDATION

Change these actions into relative update, to prevent the usage of local capture, for example, adding method increasePasscodeFailedAttempt to increate the passcode failed attempts count by 1.

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

2.4.3. Use global instead of window in React Native environment NONE

```
37  window['requestIdleCallback'] = null
```

Snippet 5. app/services/services.ts

window is polyfilled by global only for web-compatible code as in the following part of source from React Native library *https://github.com/facebook/react-*

native/blob/e60ad0837e16389a5c71b2189360278c2ee9364c/Libraries/Core/setUpGlobals.js#L21:

```
21  if (global.window === undefined) {
22    // $FlowFixMe[cannot-write]
23    global.window = global;
24  }
```

Snippet 6. React Native's setUpGlobals.js

RECOMMENDATION

Consider using global to have a more React-Native-alike environment, also move that to a global file (*global-overrides.ts* for example) so that anyone can acknowledge the modification.

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

2.4.4. Rooting

On a rooted device, any applications with the root permission can read other app data. Ronin Mobile Wallet uses Async Storage (which store in application data folder) to store data including user's mnemonic. It also uses user's passcode (which store in keychain/keystore) to encrypt the data. The passcode is stored securely but it's only 6 digits. An application with root permission can read the Ronin Mobile Wallet data folder, get the decrypted data and brute force 6-digit password to recover mnemonic.

RECOMMENDATION

- Use Keychain/KeyStore (with hardware-backed storage mechanisms if applicable) to securely encrypt/decrypt user's mnemonic. You can protect keys stored in the Keychain/KeyStore with user authentication in a confirm credential flow. The user's lock screen credentials (pattern, PIN, password, or fingerprint) are used for authentication.
- Prevent/warning users from running the wallet on rooted/emulation devices.

UPDATES

- *Mar 18, 2022*: This issue has been acknowledged and fixed by the Sky Mavis team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Sep 28, 2021*	Private Report	Verichains Lab
1.1	*Mar 18, 2022*	Public Report	Verichains Lab

Table 3. Report versions history