Urbit Keygen JS Library

Code Review

September 20th 2018

Version 1.0.0

Prepared by

Bloctrax

Introduction

This document includes the results of the code review for Urbit's Keygen JS library as found in the section titled 'Source Code'. The code review was performed by the Bloctrax team from September 12th 2018 to September 20th 2018.

The purpose of this engagement is to review Urbit's Keygen JS library source code, and provide feedback on the design, architecture, and quality of the source code.

Overall Assessment

Our overall assessment of the Urbit Keygen library is that is well written and well documented.

With that said, there are a number of small code quality and documentation issues (described issues section below) that we believe should be addressed.

In addition, there are two important ways in which the implementation of the key generation deviates from the Urbit wallet specification, which the Urbit team should address by changing either the specification or the implementation so that they are congruent with each other. This is also documented in more detail below.

Finally, while the statement coverage of the test suite is high (95.24%), the branch coverage is lower than what we would expect (81.25%) from such a foundational library. We also reviewed the test suite manually and found it to be lacking. For example, the "full wallet from seed" tests never run with more than a single ship, and the tests only assert on the network portion of the result which is only a small portion of the final result. We recommend that the library authors spend more time thinking about all the ways this library will be used and implement a more comprehensive test suite.

Specification

Our understanding of the specification was based on the following sources:

- Our understanding of the desired behavior based on our previous review of the Urbit Constitution Solidity code.
- Discussions with the Urbit team (in person and via email).
- The Urbit Wallet Specification document which was provided to us by the Urbit team.

Source Code

The following source code was reviewed:

Repository	Commit
index.js	57dff4db906950f73c52f5f3e0eb9165787e1a87

Note: This document contains a review only of the code contained in the file listed above.

Severity Level Reference

Level	Description

High	The issue poses existential risk to the project, and the issue identified could lead to massive financial or reputational repercussions.
Medium	The potential risk is large, but there is some ambiguity surrounding whether or not the issue would practically manifest.
Low	The risk is small, unlikely, or not relevant to the project in a meaningful way.
Code Quality	The issue identified does not pose any obvious risk, but fixing it would improve overall code quality, conform to recommended best practices, and perhaps lead to fewer development issues in the future.

Issues Descriptions and Recommendations

Deviations from the Urbit wallet specification

MEDIUM

The implementation of the Keygen library deviates from the Urbit wallet specification document in a few instances.

The first instance is in the `childNodeFromSeed` function. It returns the seed in a human readable hexadecimal format which we think is okay, but it also passes the

childSeed to the `walletFromSeed` function as a hexadecimal string instead of a buffer, which is problematic because the `walletFromSeed` function will call hash on that string. This will work, but the result will not be the same as hashing the underlying seed directly. In other words, SHA-512(buffer) != SHA-512(buf2hex(buffer)).

We were able to demonstrate this using the follow code snippet:

```
const keygen = require('./src/index.js');

var buf = new Buffer(10);
var hex = keygen._buf2hex(buf);

(async function main() {
  console.log("hashed hex: ", keygen._buf2hex(await
keygen._hash(hex)));
  console.log("hashed buf: ", keygen._buf2hex(await
keygen._hash(buf)));
})()
```

Another deviation from the spec is that according to the Urbit wallet specification document, the Ownership seed is supposed to be generated as: ticket → Argon2u → BIP32 but the Keygen library implementation does: ticket → Argon2u → SHA-512 → BIP32.

Web browser compatibility

CODE QUALITY

Our understanding is that the Urbit Keygen library is designed to be isomorphic (i.e. it should work both in the browser as well as in Node), however, there are a few issues with this:

1. The `Buffer` type (used in some functions like `bufferFrom`, `bufferConcat`, etc.) is not available in the web environment.

2. The isomorphic-webcrypto dependency (which appears to be a polyfill for the webcrypto API) seems like overkill since the library is already using features that won't be available in older browsers like the `async/await` keywords.

Extraneous functions

CODE QUALITY

The library defines several functions which are just one line wrappers around built-in functions. Renaming existing built-in functions hurts readability since reviewers will be familiar with the built in functions, but not the library-defined ones even though they do the same thing. Consider removing the following functions:

1. `bufferFrom`
2. `bufferConcat`
3. `reverse`

Documentation issues

CODE QUALITY

There are several minor issues with the documentation:

1. In all function parameters the types `Array` and `Buffer` should be capitalized.
2. The `hash` function documentation should state it can also accept strings.
3. The documentation for the `argon2u` function says it can accept `entropy` as a `Buffer` but doesn't include it in the type signature.
4. The documentation for the `childNodeFromSeed` function states that revision is an object, but it is an integer.
5. The `childNodeFromSeed` function documentation states that it returns a buffer, but actually it returns an object with several fields (node).
6. The `walletFromSeed` function documentation states that returns a promise which resolves to an object, but actually it just returns an object.
7. The `walletFromSeed` function documentation states that the seed should be passed as a buffer but actually it is passed as a string.
8. The boot argument for the `fullWalletFromTicket` function is undocumented.

Forked Web Argon2 Implementation

CODE QUALITY

The Keygen library appears to be using a [forked version of the argon2-browser library](). This library has two issues:

- The README on the Github page demonstrates incorrect usage of the library because it passes the password field in the options object under the “password” key when the library is expecting it under the “pass” key. When used incorrectly (as demonstrated in the README) the library will fallback to the default password which is not the desired behavior.
    - We only flagged this as a code quality issue because the Keygen library uses the “pass” field correctly.
- It’s not clear to us how the magic number 10 (used in the Keygen library) maps to argon2.ArgonType.Argon2u. In addition, the Argon2 library that Urbit’s fork is derived from exports the different Argon types (but it does not export Argon2u) which seems like a better solution than passing around magic numbers.

Appendix

Exhibit A -Disclaimer

Bloctrax makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and Bloctrax specifically disclaims all implied warranties or merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

Bloctrax will not be liable for any lost profits, business, contracts, revenue, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand against company by any other party. If no event will Bloctrax be liable for consequential, incidental, special, indirect, or exemplary damages

arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if Bloctrax has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the Urbit team and only the source code Bloctrax notes as being within the scope of Bloctrax's review within this report. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute quality or security of the project.

In this report you may through hypertext or other computer links, gain access to websites operated by persons other than Bloctrax. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such websites' owners. You agree that Bloctrax is not responsible for the content or operation of such websites, and that Bloctrax shall have no liability to your or any other person or entity for the use of third party websites. Bloctrax assumes no responsibility for the use of third party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.