micro-stacks Audit

April 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the micro-stacks project. The package serves Typescript projects building applications in the Stacks ecosystem.

First we will provide a summary of our discoveries and then we will show the details of our findings.

Scope

The code audited are from the https://github.com/fungible-systems/micro-stacks/ git repository. The audit is based on the commit `651d4cc699c02e684954a13f176812502fbf5db1`.

Summary of Findings

We found a critical issue, three medium issues and two minor issues. Also, several enhancements were proposed.

ID	Title	Severity	Status
CR-01	Import from HTTPS May Lead To Unwanted Code	Critical	Unresolved
ME-01	Using Unaudited Cryptography Primitives	Medium	Unresolved
ME-02	Using Code For Which the Audited Version Is Unclear	Medium	Unresolved
ME-03	Replay Attack in Elliptic Curve Signature	Medium	Unresolved
MI-01	Use of Unsafe Comparisons	Minor	Unresolved
MI-02	Conversion Defaulting to Unsafe Value When Undefined	Minor	Unresolved

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

CR-01 Import from HTTPS May Lead To Unwanted Code

Location:

- `crypto-extras/crypto-aes/mod.ts:3-4`

```
import { Aes } from 'https://deno.land/x/crypto/aes.ts';
import { Cbc, Padding } from 'https://deno.land/x/crypto/block-modes.ts';
```

Importing modules from an https location is dangerous as the site could be compromised or the code could be changed at any time.

Recommendation
Always make imports from locations ensuring availability and integrity, like the npm (with package-lock or yarn.lock) or standard git repositories and using a commit hash. The alternative, which should always come if the former option is unavailable, is to copy the files to your repository.

Status
Unresolved.

Medium Severity Issues

ME-01 Using Unaudited Cryptography Primitives

The AES implementations imported from deno.land have not been audited, and as such are liable to errors.

```
import { Aes } from 'https://deno.land/x/crypto/aes.ts';
import { Cbc, Padding } from 'https://deno.land/x/crypto/block-modes.ts';
```

Moreover, there are no tests for src/crypto-extras/crypto-aes.ts .

Recommendation
Always make imports from locations ensuring availability and integrity, like the npm (with package-lock or yarn.lock) or standard git repositories using a commit hash. The alternative, which should always come if the former option is unavailable, is to copy the files to your repository and documenting this.

Include tests for these functions using, for example, NIST tests vectors.

Status
Unresolved.

ME-02 Using Code For Which the Audited Version Is Unclear

The imports from @noble libraries refer to specific versions of these:

```
"@noble/hashes": "1.0.0",
"@noble/secp256k1": "1.5.5"
```

And these libraries have been audited (https://github.com/paulmillr/noble-secp256k1#security and https://github.com/paulmillr/noble-hashes#security), yet the audits specify no commit hash or specific version of the code and it is therefore not possible to determine what changes occurred between audited and imported versions.

Recommendation
Ensure using the audited version.

Status
Unresolved.

ME-03 Replay Attack in Elliptic Curve Signature

The function `signECDSA()` in `crypto/encryption/sign.ts` and the function `createWithSignedHash()` in `crypto/token-signer/token-signer.ts` do not force the signature to be unique. This allows for replay attacks, as any attacker that gets a signature generated by these functions can generate a different signature for the same message. This could be used to compromise a system in combination with other vulnerabilities.

Respectively, the verification functions in `/crypto/token-signer/token-verifier.ts` allow this replay attack when `strict=false`.

Recommendation
Insecure options should be removed. If absolutely necessary, then first make the default behavior secure and also warn users when opting for the insecure parameter option.

Status
Unresolved.

Minor Severity Issues

MI-01 Use of Unsafe Comparisons

Location:

- `src/api/utils.ts:65`
- `src/transactions/authorization.ts:128`

- `src/transactions/authorization.ts:137`
- `src/transactions/contract-abi.ts:229`
- `src/transactions/contract-abi.ts:234`
- `src/transactions/common/utils.ts:6`

Throughout the code there are several instances where comparisons are made via the == operator instead of the standard === operator. Note that this may lead to unsafe comparisons.

Recommendation
Replace == with === and test.

Status
Unresolved.

MI-02 Conversion Defaulting to Unsafe Value When Undefined

Location:

- src/crypto/c32/index.ts:61-63
- src/crypto/c32/index.ts:78-80

The function `b58ToC32()` (respectively `c32ToB58()`) lookup the `stacksVersion` (`bitcoinVersion` respectively) in the key-value `BITCOIN_TO_STACKS_NETWORK_VERSION[addrVersion]` (resp. `STACKS_TO_BITCOIN_NETWORK_VERSION[addrVersion]`). When this returns undefined, a value is set from `addrVersion`. This value could be incorrect and cause unforeseen problems with a security impact.

Recommendation
If the value `stacksVersion` or `bitcoinVersion` value returned is undefined, then throw an exception and allow the developer to fix the error.

Status
Unresolved.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	Inconsistent Constant Declarations	Not implemented
EN-02	Best Practices With TODO Messages	Not implemented
EN-03	Test Code in Production	Not implemented
EN-04	Token Expiration Insecure by Default	Not implemented
EN-05	Documentation Details	Not implemented
EN-06	References to Git in Production Code	Not implemented

EN-01 Inconsistent Constant Declarations

The constant declaration

```
const BITCOIN_VERSIONS: Versions = { private: 0x0488ade4, public:
0x0488b21e };
```

Is used at least in two places in the code (`src/crypto/base58/networks.ts` and `src/crypto/base58/networks.ts`). Constants should be defined once and in a specific place so that they become easier to manage and in order to prevent inconsistencies.

EN-02 Best Practices With TODO Messages

Throughout the code there are several comments which are not part of the documentation and should be removed. These may confuse package users.

Status

Not implemented.

EN-03 Test Code in Production

There is code (an if statement) in line 11 of `src/transactions/fetchers/broadcast-transactions.ts` which is specific to a test yet is part of the production code.

Recommendation

Remove this code and move testing code to test files.

EN-04 Token Expiration Insecure by Default

The function `makeAuthResponse()` in `src/wallet-sdk/auth/index.ts` has an optional expiresAt parameter, which is set to one month by default. This could be overly and insecure in certain contexts.

Recommendation

Make the default smaller, e.g., 5 minutes.

EN-05 Documentation Details

Throughout the code there are several places in the comments that require attention.

- Line 53 in the file src/crypto/c32/index.ts goes like this

  ```
  * @param version - the version number, if not inferred from the address
  ```

  Yet this is not a function parameter and the line should be deleted.

- Similarly, the comment in `src/crypto/encryption/sign.ts:5-15` belongs to the function `signECDSA()` and should be placed immediately before this function.
- The comment in lines 25-29 of `src/crypto/common/ecies-utils.ts` documents a function elsewhere in the code, probably `eciesGetJsonStringLength()` that happens in line 97

EN-06 References to Git in Production Code

At different places in the code there are references to other projects via their repositories but omitting hashes and versions. This could lead to mismatches. For example, `src/crypto/base58/index.ts:6` points to code no longer available (master version has changed), and the same happens with the following:

- `src/clarity/abi.ts:1`
- `src/transactions/types.ts:143`
- `src/wallet-sdk/constants.ts:5`

Changelog

- 2022-04-25 – Initial report based on commit `651d4cc699c02e684954a13f176812502fbf5db1`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the micro-stacks project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.