AllBridge Audit

June 2023

By CoinFabrik

v202306

Executive Summary **3**
Scope **4**
Contracts Descriptions **4**
Methodology **5**
Findings **6**
Severity Classification 6
Issues Status 6
Critical Severity Issues 6
Medium Severity Issues 7
ME-01 Race Condition On Fees Charge 7
ME-02 Insecure Authentication Through tx-sender 7
Minor Severity Issues 9
MI-01 Internal Error Handled In Outer Context 9
Enhancements 10
EN-01 Unnecessary Check Against Primitive true 10
Other Considerations **11**
Centralization 11
Upgrades 11
Changelog **12**

Executive Summary

CoinFabrik was asked to audit the contracts for the AllBridge project.

Allbridge serves as a platform that facilitates the connection between different blockchain networks. It links EVM-compatible blockchains, such as Ethereum, Polygon, and BSC, with non-EVM blockchains, namely Solana, Terra, and the newly added, Stacks.

During this audit we found no critical issues, two medium issues and one minor issue. Also, an enhancement was proposed.

In the following table we summarize the security issues we found in this audit. The severity classification criteria and the status meaning are explained below. This table does not include the enhancements we suggest to implement, which are described in a specific section after the security issues.

ID	Title	Severity	Status
ME-01	Race Condition On Fees Charge	Medium	Acknowledged
ME-02	Insecure Authentication Through tx-sender	Medium	Resolved
MI-01	Internal Error Handled In Outer Context	Minor	Resolved

Scope

The audited files are from the git repository located at https://github.com/allbridge-io/bridge-stacks-contract-public. The audit is based on the commit `cad07c5e1f46ce17f6bd3eee2bf268a4be58f85f`. The fixes were checked on commit `21470cfd5aa70088a9bcbe8b281b8fab37c89b25`

The scope for this audit includes and is limited to the following files:

- `contracts/bridge.clar`: Bridge endpoint.
- `contracts/tokens/i-stx.clar`: Sip-010-compliant wrapper for STX.
- `contracts/tokens/n-token.clar`: Token SIP-010.
- `contracts/tokens/w-token.clar`: Wrapped token SIP-010.

No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Contracts Descriptions

`contracts/bridge.clar`

This is a bridge contract that facilitates the transfer of tokens from or to Stacks. Here is a breakdown of the public functions of the contract:

- `add-token`: This function is used to add a token to the bridge. It validates the token information and saves it in the 'assets' map.
- `remove-token`: This function is used to remove a token from the bridge. It validates ownership and removes the token from the 'assets' map.
- `lock`: This function locks tokens on the bridge. It validates the input, creates a lock, transfers fees, and transfers tokens to the bridge.
- `unlock`: This function unlocks tokens on the bridge. It validates the input, creates an unlock, and transfers tokens to the recipient.

The contract aims to provide a secure and standardized way of transferring tokens between different chains through the bridge functionality. It supports adding and removing tokens, locking and unlocking tokens, and handles fees and ownership transfers.

Methodology

CoinFabrik was provided with the source code, including automated tests that define the expected behavior, and general documentation about the project. Our auditors spent one weeks auditing the source code provided, which includes understanding the context of use, analyzing the boundaries of the expected behavior of each contract and function, understanding the implementation by the development team (including dependencies beyond the scope to be audited) and identifying possible situations in which the code allows the caller to reach a state that exposes some vulnerability. Without being limited to them, the audit process included the following analyses.

- Arithmetic errors
- Race conditions
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

After delivering a report with our findings, the development team had the opportunity to comment on every finding and fix the issues they considered convenient. Once fixed and/or commented, our team ran a second review process to verify that the changes to the code effectively solve the issues found and do not unintentionally add new ones. This report includes the final status after the second review.

Findings

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. Blocking bugs are also included in this category. They must be fixed **immediately**.
- **High:** These refer to a vulnerability that, if exploited, could have a substantial impact, but requires a more extensive setup or effort compared to critical issues. These pose a significant risk and demand immediate attention.
- **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 might 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 has one of the following statuses:

- **Unresolved**: The issue has not been addressed.
- **Acknowledged**: The issue remains in the code, but is a result of an intentional decision. The reported risk is accepted by the development team.
- **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

No issues found.

Medium Severity Issues

ME-01 Race Condition On Fees Charge

Location:

- `contracts/bridge.clar: 176`

The owner can set any percentage as a fee at any time, even after the user signed the transaction and before it is executed. There is no maximum value enforced in the setter. In order to exploit it, the owner needs to front-run the call to the `lock()` function and set the minimum fee for the token or the base fee rate to any value, higher than the one accepted by the user. Consequently, the fee collector gets all the tokens.

This issue can only be exploited by the owner, or someone impersonating the owner. The end users of this bridge are the ones impacted by this risk.

Steps to replicate

Initially, fees are setted to 5%.

- User A sends a transaction calling the lock function.
- Owner frontruns that transaction, paying a higher gas price, setting the fee to 100%.
- Owner's transaction gets executed and the fee is modified.
- User A's transaction is executed, but zero amount is bridged and all the value is transferred to the fee collector address.

Recommendation

It could be solved by making the user provide the maximum fee accepted as an argument to the `lock()` function. This is the safer solution, since the user has control over the accepted fee.

Otherwise, the risk could be mitigated enforcing the fee to be lower than a constant maximum value. This solution is simpler, but still allows the owner to set a fee higher than the one accepted by the user.

Status

Acknowledged. The development team decided not to fix this issue.

ME-02 Insecure Authentication Through tx-sender

Location:

- `contracts/bridge.clar`
- `contracts/tokens/i-stx.clar`
- `contracts/tokens/n-token.clar`
- `contracts/tokens/w-token.clar`

Using `tx-sender` for authentication is not secure. Actors in the system could be targeted for phishing. An attacker could trick these actors to call a malicious contract which then internally calls one of the following functions. As a consequence, the actor is impersonated by the attacker.

This issue was found in:

- `contracts/bridge.clar:`
  - `add-token()`
  - `remove-token()`
  - `set-contract-owner()`
  - `set-base-fee-rate-bp()`
  - `set-fee-collector()`
  - `set-validator-public-key()`
  - `set-is-bridge-enabled()`
  - `set-token-min-fee()`
- `contracts/tokens/i-stx.clar:`
  - `set-contract-owner()`
  - `set-token-uri()`
- `contracts/tokens/n-token.clar:`
  - `set-contract-owner()`
  - `set-precision()`
  - `transfer()`
- `contracts/tokens/w-token.clar:`
  - `set-contract-owner()`
  - `set-token-uri()`
  - `set-token-name()`
  - `set-token-symbol()`
  - `transfer()`

Functions that involve asset transfers could not be called in the attack, thanks to proper use of post-conditions.

Steps to replicate

- Attacker creates a malicious contract which calls the `set-contract-owner()` function with the attacker principal as the new owner.
- The attacker deceives the current owner to call the malicious contract.
- The owner calls the malicious contract. Then, the owner is changed.
- Now, the attacker can call any function restricted to the owner. For instance, can execute the exploit described in ME-02.

Recommendation

Prefer `contract-caller` to `tx-sender` for authentication. Also, adding a mapping for trusted callers might be helpful if specific intermediary contracts are needed.

Status
Resolved. The first recommendation was implemented.

Minor Severity Issues

MI-01 Internal Error Handled In Outer Context

Location:

- `contracts/bridge.clar: 247, 296, 482`

Internal errors, such as those that can be triggered when calling `map-set()`, should be handled by the immediate function calling it.

For instance, `lock()` function calls internally to `create-lock()`. `create-lock()` returns the response from a `map-set()` wrapped in an ok. Then, the `lock()` function has to verify the value wrapped is not an `(ok false)`, which implies an internal error. This error needs to be always verified, since a `false` would mean the record was not written. Because of this, every function calling `create-lock()` has to handle that exception.

In this context, it does not suppose a risk since no function in scope ignores the wrapped boolean. However, new functions might not verify that value and just ensure naively the response is not error.

Recommendation
Consider handling exceptions immediately instead of doing it in an outer context. Replace `(ok (map-set M k v))` with `(asserts! (map-set M k v) err-internal-storage)` and `(ok true)` at the end.

Status
Resolved. Fixed according to recommendation.

Enhancements

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

ID	Title	Status
EN-01	Unnecessary Check Against Primitive true	Implemented

EN-01 Unnecessary Check Against Primitive true

Booleans values do not need to be checked against the primitive `true` using `is-eq()`, since the result will always be the same as the original value.

This checks can be found in:

- `save-token()`
- `remove-token()`
- `clean-assets-maps()`
- `assert-token-type()`
- `assert-precision()`
- `assert-min-fee()`
- `assert-lock-input()`

Recommendation

Remove these checks in order to simplify the codebase and reduce the runtime cost.

Status

Implemented.

Other Considerations

The considerations stated in this section are not right or wrong. We do not suggest any action to fix them. But we consider that they may be of interest to other stakeholders of the project, including users of the audited contracts, token holders or project investors.

Centralization

The system relies on an off-chain validator which is responsible for signing the cross-chain transfers and the bridge contract has an owner which can change the registered public key for the validator, the fees and stop the system by pausing the entire contract or disabling specific tokens.

Upgrades

The contracts do not have any mechanism for upgrading them in a future time.

Changelog

- 2023-06-16 – Initial report based on commit `cad07c5e1f46ce17f6bd3eee2bf268a4be58f85f`.
- 2023-07-18 – Final report based on commit `21470cfd5aa70088a9bcbe8b281b8fab37c89b25`.

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