ALEX Audit

Bridge backend and endpoints

April 2023

By CoinFabrik

v202209

Executive Summary

CoinFabrik was asked to audit the contracts and the back-end for the ALEX bridge's project. The audited files are from the git repository located at https://github.com/alexgo-io/alex-cloud/. The audit is based on the commit `e2372e68a19a4f7cbcee75c46ad6d94c5c420e80`.

This project is a Stacks-Ethereum hybrid bridge which allows users to transfer their assets across those blockchains.

The whole bridge project consists of the audited endpoints, an interaction with the Wrapped vendor, and off-chain components that monitor on-chain transactions and approve the asset transfer.

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

Two issues were resolved and two were mitigated. The enhancement was not implemented.

Scope

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

- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar`: Stacks endpoint. Users send wrapped tokens in order to unwrap and withdraw them on Ethereum. Also, it holds the wrapped tokens for those who bridge from its Ethereum counterpart until a relayer executes the transfer.
- `packages/contracts/bridge-solidity/contracts/BridgeEndpoint.sol`: Ethereum endpoint. Users send unwrapped tokens in order to wrap and bridge them to Stacks. This endpoint holds unwrapped tokens for transferring to those who bridge from Stacks, unwrapping their tokens.
- `packages/contracts/bridge-solidity/contracts/utils/ERC20Fixed.sol`: Helper library for 18-decimal fixed point precision.
- `packages/contracts/bridge-solidity/contracts/utils/Allowlistable.sol`: Library for users whitelist.
- `packages/contracts/bridge-solidity/contracts/utils/Errors.sol:` Library which defines custom error messages for the endpoint contract.
- `packages/contracts/bridge-solidity/contracts/utils/math/FixedPoint.sol:` Math library for gas-efficient fixed-point operation.
- `packages/contracts/bridge-solidity/contracts/utils/math/LogExpMath.sol:` Exponentiation and logarithm functions for 18 decimal fixed point numbers.

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.

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 four 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
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- 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

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	Hardcoded Credentials on Api-Server	Medium	Resolved
MI-01	Insecure Authentication Through tx-sender	Minor	Mitigated
MI-02	Fee Higher Than Expected	Minor	Mitigated
MI-03	Unchecked TX proof on stacks-relayer	Minor	Resolved

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 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 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

No issues found.

Medium Severity Issues

ME-01 Hardcoded Credentials on Api-Server

Location:

- `./packages/apps/api-server/src/app/api/guards/auth.guard.ts:4`

Credentials of several API users are hardcoded inside the source code.

Recommendation

Load credentials from an external file or environment variable. If possible, don't commit test-credentials as they are very often shipped and activated in the final production deployment by mistake.

Status

Resolved. Fixed according to the recommendation.

Minor Severity Issues

MI-01 Insecure Authentication Through tx-sender

Location:

- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar: 84, 218, 383`

Using `tx-sender` for authentication is not secure. Actors in the system could be targeted for phishing.

This issue was found in:

- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar::transfer-to-wrap()`
- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar::check-is-owner()`
- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar::transfer-to-unwrap()`

Functions that involve asset transfers cannot be called in the attack, thanks to proper use of post-conditions. Also, owner authentication will be less likely to be targeted once the DAO is set as owner.

Recommendation

Prefer `contract-caller` to `tx-sender` for authentication, unless it is specifically required and the risk is considered. Also, adding a mapping for trusted callers might be helpful if intermediary contracts are needed.

Status

Mitigated. The development team stated it is mitigated for owner authentication given that all these contracts will be owned by a DAO. For other cases, given that post-conditions can prevent asset transfers triggered by malicious actors through `tx-sender`, they are in favor of keeping `tx-sender` in place for composability.

MI-02 Fee Higher Than Expected

Location:

- `packages/contracts/bridge-solidity/contracts/BridgeEndpoint.sol: 115-149,`
- `packages/contracts/bridge-stacks/contracts/bridge-endpoint.clar: 84-121`

The fee charged to the user is calculated based on the minFee and feePerToken variables. If a user sends a transaction in order to wrap their tokens and those variables are increased before the transaction is processed, the fee charged will be higher than what the user accepted to pay.

These variables could be set to 100% of the total amount wrapped and the user will not receive anything in exchange. However, this is mitigated since the contract owner will be a governance contract and such a change requires a proposal process.

Recommendation

Add a parameter for the user to set the maximum accepted fee to the functions which charges fees and revert when the charged fee is greater than this parameter.

Status

Mitigated. The development team stated it is mitigated given that all these contracts will be owned by a DAO. Therefore, changes require a proposal process.

MI-03 Unchecked TX Proof on Stacks-Relayer

Location:

- `./packages/apps/stacks-relayer/src/main.ts: 74`

In the function `submitOnChainProof()` when no proof of the TX is found, it just logs a warning and returns:

```
const p0 = pendingProofs[0];
if (p0 == null) {
  logger.warn(`No proof found for ${source_transaction_unique_id}`);
  return
```

In contrast, the ethereum-relayer application, throws an assert() when detecting this condition (`ethereum-relayer/src/main.ts:61`):

```
const p0 = pendingProofs[0];
assert(p0, `No proof found for ${source_transaction_unique_id}`);
```

This causes that a missing proof won't be detected on the stacks-relayer `submitOnChainProof()` function.

Recommendation

Throw an assertion if the lack of proof is detected on the stacks-relayer application.

Status

Resolved. Fixed according to the recommendation.

Enhancements

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

ID	Title	Status
EN-01	Prefer ApiExcludeController To Excluding Every Endpoint	Not implemented

EN-01 Prefer ApiExcludeController To Excluding Every Endpoint

Location:

- `packages/apps/api-server/src/api/admin/validator.controller.ts:22-168`,
- `packages/apps/api-server/src/api/controllers/index.controller.ts:4-17`

Instead of excluding every endpoint in a controller, exclude the whole controller, so that new endpoints added are already excluded by default.

Recommendation

Add the `@ApiExcludeController()` decorator to the controller and remove the `@ApiExcludeEndpoint()` decorator from each function.

Status

Not implemented. The development team will consider including this to the backlog.

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

Since this bridge project took a hybrid approach, there are unavoidable centralization points:

- Endpoints depend on an intermediary process in order to work.
- A relayer sends a transaction in order to execute token transfers.
- The relayer has to provide validator signatures.
- A contract owner defines the required amount of signatures, the approved recipients and the relayers.

Upgrades

The audited contracts do not provide mechanisms for eventual upgrades, while the back-end is, as expected, mutable and can be upgraded at any time.

Privileged Roles

These are the privileged roles that we identified on each of the audited contracts.

BridgeEndpoint.sol and bridge-endpoint.clar

Owner

Defines system parameters like designated validators, relayers, signatures requirement, and approved tokens. The owner can withdraw tokens from the endpoint at any time.

Validator

Verifies bridge transactions and approves them providing a signature with the order hash to a relayer.

Relayer

Completes bridges by providing validator signatures in a call to the endpoint where the tokens are transferred to the user.

Changelog

- 2022-04-05 – Initial report based on commit `e2372e68a19a4f7cbcee75c46ad6d94c5c420e80`.
- 2022-04-13 – Final report based on commit `0a0d8f53ff94e0abb444986ca10d5a6a63c6128e`.

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