CoinFabrik

Magic Bridge Audit

May 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the Magic Bridge project. First we will provide a summary of our discoveries and then we will show the details of our findings. After auditing the contracts, developers fixed issues and the fixes were audited; details of this revision are included.

Scope

The contracts audited are from the https://github.com/magicstx/bridge git repository. The audit is based on the commit `0fe0125735b2381e95e51456673b3e10e03f0a2b`. Revision is based on commit hash `9a9e7ca2b4555c07d9d818b65f2000332c6307b4`.

The audited contracts are:

- `contracts/bridge.clar:` Code for bridge.
- `contracts/clarity-bitcoin.clar`: Utilities for processing blocks and transactions out of the Bitcoin network.

The scope of the audit is limited to those files. 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.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- 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

Summary of Findings

We found 2 critical issues and 2 medium issues. Also, one enhancement was proposed. All issues were fixed but a critical vulnerability, which has been acknowledged. See the note in the issue's description for more details on this issue.

Security Issues

ID	Title	Severity	Status
CR-01	Wrong xBTC Address	Critical	Acknowledged
CR-02	Frontrunning Attack by Changing Fees In Inbound Swap	Critical	Fixed
ME-01	Unbounded And Undocumented Fees	Medium	Fixed
ME-02	Unfinished Transactions May Lock Funds	Medium	Fixed
ME-03	Outbound Revocation Taking Longer Than Expected	Medium	Fixed
MI-01	Unsafe Parameter May Lead To Locked Funds	Minor	Fixed

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 Wrong xBTC Address

Location:

- `contracts/bridge.clar:621,`

The transfer function makes a call to a contact xbtc.clar:

```
contract-call? .xbtc transfer amount sender recipient none
```

which is not the xBTC (external) contract, but a contract with the same name defining an homonymous token. Deploying the contracts as in this commit could lead to confusion and having unprevented users swap their BTC for this token with a few hard-coded addresses.

Recommendation

Remove the xbtc.clar contract and use the xBTC address instead.

Status

Acknowledged. Developers informed us auditors that this xbtc.clar contact is a mock and will be replaced on production pointing to the correct contract (this one).

Developers further explained that "... a new (unreleased) version of Clarinet includes support for defining external dependencies, which is exactly what we need in this use case. We will add support for this feature as soon as the new version is released."

CR-02 Frontrunning Attack by Changing Fees In Inbound Swap

A supplier may frontrun a swapper by calling `update-supplier-fees()` before the swapper calls for `escrow-swap()` raising fees indiscriminately. The swapper thus cannot ensure fees when he calls the escrow and after this moment there is no turning back.

Recommendation

Allow the swapper to establish bounds for the fees, thus preventing the supplier from modifying fees after a certain point.

Status

Fixed. A new parameter, `min-to-receive`, has been added to the `escrow-swap()` which was modified to check if the xbtc received by the swapper is at least `min-to-receive`.

Medium Severity Issues

ME-01 Unbounded And Undocumented Fees

A supplier may register via the `register-supplier()` function, also setting fees and later alter these with the `update-supplier-fees()`. None of the functions makes any check over `outbound-base-fee` and `outbound-base-fee`. Also, the call for `validate-fee()` function is made only during registration but not during fee update. Finally, there is no documentation for fees nor how many decimals are used, which could lead to errors from both supplier and swapper. Finally, the constant in `validate-fee()` should be defined in the contract's preamble and not within the code to improve readability and prevent coding errors.

Recommendation

Ensure `validate-fee()` runs when modifying inbound-fee or outbound-fee. Set maximum value for base-fees. The maximum value for (percentage) fees is set to 100% which is plausible, but it is recommended to lower this.

Status

Fixed. The `update-supplier-fees()` function was modified to run `validate-fee()` on `outbound-fee` and `inbound-fee`. With this change fees remain bounded.

ME-02 Unfinished Transactions May Lock Funds

In an inbound swap, if the swapper calls an escrow but the swap is not finalized by expiration time, then the swapper may unlock his BTC from the Bitcoin network, but the supplier cannot call `remove-funds()` or retrieve his xBTC.

If the `finalize-swap()` is performed but doesn't confirm on Stacks before the swap expires, the supplier may see the preimage in the swapper may lose their BTC without getting xBTC.

Recommendation

Allow a supplier to call `remove-funds()` after expiration or create a specific `revoke-inbound-swap()` function. Alternatively, have an admin that can revert locked transactions manually when needed.

Status

Fixed. A function, revoke-expired-inbound(txid), was added to allow revoking an expired transaction given its transaction id.

ME-03 Outbound Revocation Taking Longer Than Expected

When calling `revoke-expired-outbount()` to revoke an expired outbound swap, the function calls for `validate-outbound-revocable()` that compares the expected expiration (computed as the (burn) block height where the swap was confirmed plus a constant) with the current burn block height. However, the function is using the `block-height` keyword which looks up the Stacks block height, which is smaller (and different). This imposes a substantial delay to recover funds.

The bug was discovered by the development team and a quick fix was implemented.

Status

Fixed. The `block-height` keyword was replaced with the `burn-block-height` keyword fixing the issue.

Minor Severity Issues

MI-01 Unsafe Parameter May Lead To Locked Funds

In an inbound swap a swapper may establish an expiration date too far away into the future locking his and the supplier's funds for that time. This is unnecessary for swaps which should never take over a day.

This issue was caught by an independent audit, informed to us by the development team, and included herein for completeness.

Status

Fixed. The issue was fixed by adding a limitation to the bridge::validate-expiration() function called by ::escrow-swpa() by the swapper preventing him from locking the supplier's funds for longer than 550 blocks.

Enhancements

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

Table

ID	**Title**	**Status**
EN-01	Supplier Name Leading To Phishing Attacks	Implemented

Details

EN-01 Supplier Name Leading To Phishing Attacks

A registered supplier may change his name. A second supplier may then use this name and impersonate the first one. Alternatively, a supplier may pick a name which is very similar to another supplier. In both cases a swapper identifying a supplier via the `get-supplier-by-name()` function may fail to understand what supplier he is interacting with and fall victim to a phishing attack.

Recommendation

Remove this function unless necessary. In that case, document risks for users within the smart contact and any web interface users may deal with.

Status

Implemented. The name parameter from the supplier mappings and the `get-supplier-by-name()` function were removed.

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 for other stakeholders of the project, including users of the audited contracts, owners or project investors.

Documentation

The documentation could be improved. Documentation within the code explains the use of the most relevant functions, and so does the docs. However, there are other steps in the protocol (e.g., happening off network or in the Bitcoin network) which are not explained.

Changelog

- 2022-05-10 – Initial report based on commit `0fe0125735b2381e95e51456673b3e10e03f0a2b`.
- 2022-06-21 - Revision based on commit `9a9e7ca2b4555c07d9d818b65f2000332c6307b4`.
- 2022-06-24 - Including ME-03 bug fix on commit `6e076a7d503e61199ed7c426f1682c8c1edccdc9`.