Quantstamp

Rhino.Fi

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Token Bridge
Timeline	2024-09-06 through 2024-09-06
Language	FunC
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Readme
Source Code	• rhinofi/contracts_public #ce58703
Auditors	• Jonathan Mevs Auditing Engineer • Julio Aguilar Auditing Engineer • Cameron Biniamow Auditing Engineer

Documentation quality	Low
Test quality	Medium
Total Findings	3 Fixed: 2 Acknowledged: 1
High severity findings	0
Medium severity findings	0
Low severity findings	1 Fixed: 1
Undetermined severity findings	0
Informational findings	2 Fixed: 1 Acknowledged: 1

Summary of Findings

Rhino.fi is a DeFi platform with cross-chain capabilities that allow users to bridge funds between various blockchains. Rhino Fi's bridge design is based on collateralized liquidity pools across chains, rather than the common "burn and mint" model. This method ensures quick transfers and easy onboarding of new chains. Security is maintained as Rhino Fi itself collateralizes the funds.

This audit revolves around the Rhino.fi cross-chain bridge developed for the TON blockchain. Specifically, only the `bridge_contract.fc` was in scope. Therefore, all off-chain components, such as message passing and message verification from other blockchains, are out of the scope of this audit.

During the review, the audit team discovered high-quality code and a sufficient test suite in the codebase. However, the audit team recommends improving overall documentation and adding end-to-end test cases covering the bridging process from one blockchain to another.

One low severity issue describes the potential for lost jettons during deposit due to user error. Additionally, this report includes two informational issues and auditor suggestions to ensure adherence to best practices. The Rhino.fi team was very collaborative throughout the audit.

Fix Review Update

All issues in the report have either been fixed or acknowledged with sufficient reasoning and tests have been updated to test new features. During the fix review, all changes were merged to the public repo mentioned throughout this report. We note here that the initial audit reviewed code on a private repo and we confirm that the merges to the public repo are identical to what we reviewed.

ID	DESCRIPTION	SEVERITY	STATUS
RHNO-1	Potential for Lost Funds in Jetton Deposit	Low	Fixed
RHNO-2	Bounceable Message Sent Despite No Processing of Such	Informational	Acknowledged
RHNO-3	Missing Input Validation	Informational	Fixed

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

> **Disclaimer**
>
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Repo: https://github.com/rhinofi/contracts_public(ce587034645e4482b680990fffeb3871e6dbf458)
Files: contracts/* , contracts/imports/*

Files Excluded

Repo: https://github.com/rhinofi/contracts_public(ce587034645e4482b680990fffeb3871e6dbf458)
Files: contracts/jetton/*

Operational Considerations

1. Due to the high degree of centralization, users must trust the Rhino.fi authorized addresses will act honestly to fulfill bridge requests between different blockchains.
2. The Rhino Fi team should be especially cautious when transferring ownership and upgrading the contract with new data to be set. In the case where the owner is assigned to an address that is not confirmed by the team, the team will lose control of this contract and all funds within.

Key Actors And Their Capabilities

The Rhino.fi TON bridge is highly centralized and requires trusted key actors to initiate the movement of TON and Jettons between the bridge contract and user wallets. The following list states each key actor and their capabilities in the system.

1. `owner`
    1. Can add or remove an authorized address to the `bridge_contract`. See "Authorized Addresses" for a list of their capabilities.
    2. Can transfer ownership of the `bridge_contract` to another address.
    3. Can upgrade the code and state of the `bridge_contract`.

2. Authorized Addresses
    1. Can enable or disable global deposits.
    2. Can set the deposit limit for any given address.
    3. Can withdraw any amount of Jettons from the `bridge_contract`.
    4. Can withdraw any amount of TON from the `bridge_contract`.
3. Whitelisted Jetton Wallets
    1. Can deposit Jettons into the `bridge_contract` up to the deposit limit when global deposits are enabled.
4. Users
    1. Can deposit TON into the `bridge_contract` up to the deposit limit when global deposits are enabled.

Findings

RHNO-1 Potential for Lost Funds in Jetton Deposit

Low Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `f219cb4c248a6ff7ad29046acbc674169c2fb42d`.
The client provided the following explanation:

```
The provided fix checks now that the forward_payload length is 160 bit and that the token has been
whitelisted, otherwise the Jetton return flow is initiated by default.
  Is not advisable for users to interact directly with the contract instead of using the UI provided by
Rhino, which will provide the necessary checks and will ensure that the transaction is built properly.
```

File(s) affected: `bridge_contract.fc`

Description: A user's jetton deposit begins from their jetton wallet, sending a message to the Rhino bridge's jetton wallet to begin a deposit on their behalf. The user is expected to specify an EVM address in the `forward_payload` of this message, which represents the destination EVM wallet to which they are bridging funds to. However, if the user sends funds and a message while accidentally forgetting to include the `forward_payload`, this user's funds will be stuck in the Rhino's bridge wallet with no way of recovery besides a manual operation by the Rhino team.

```
if(forward_payload.slice_empty?()){
    return();
}
```

Recommendation: Instead of ignoring messages with an empty `forward_payload`, the original depositor should have their jettons refunded. The `jetton_sender` extracted from the `msg_body` is the original sender of the jettons. if they mistakenly forgot to specify the `forward_payload` that includes the target EVM address, they should have their jettons returned.

RHNO-2 Bounceable Message Sent Despite No Processing of Such

Informational Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
send_jettons() is only used for withdrawals and for the return Jetton flow.
  Withdrawals are only actioned by authorized users (RhinoFi backend broadcasters).
  The Jetton return flow is actioned when users directly interact with the contract when deposits are
disabled.
  There's no way to safely handle bounced Jetton messages on-chain. The most probable way to have a
bounced transaction when sending Jettons from the contract is by not attaching enough gas in the Jetton
Notify message.
  If that's the case, it could be possible that there's not enough gas to even send a bounced message
in the first place, and even if there was enough gas, the action could not be retried since the amount
of gas left after the bounce would be even lower.
  RhinoFi provides backend checks for failed transactions to to be retried in case of failure.
  Morever, RhinoFi provides a UI to build and submit the transactions that prevents the return Jetton
flow to incur, an ensures that the transaction will have enough gas and the correct Jetton parameters
to succeed.
  Additionally, RhinoFi scans failed returned Jetton transactions in the backend to alert if a return
Jetton flow failed so it can be actioned internally.
```

File(s) affected: `jetton_utils.fc`, `bridge_contract.fc`

Description: `send_jetton()` sends a bounceable message despite there being no processing for a bounced message in the contract.

Recommendation: We recommend still sending the jettons as a bounceable message, however, there needs to be logic included for processing a bounced message.

RHNO-3 Missing Input Validation

Informational | Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `081d898c91ae2458b9fa02f1886cb83400f44ebc`.
The client provided the following explanation:

```
1. Data validation has been introduced in the code. The owner of the contract is not allowed to be
   changed during an upgrade operation, ensuring that the only way to transfer the ownership is through
   the transfer_ownership/accept_ownership mechanism. That's the safest way to ensure the owner will be
   always valid.
2. set_deposit_limit can be 0 when deposits have been disabled for a specific whitelisted token
3. Now it's impossible to change the owner during an upgrade operation.
```

File(s) affected: `upgrade.fc`

Description: It is important to validate inputs to avoid human error, even if they only come from trusted addresses:

1. Fixed In `upgrade.upgrade_contract_code_and_data()`, `data` should be validated to ensure the `owner` address is valid. Further, `code` should be validated to be non-zero.
2. Acknowledged When processing `op::set_deposit_limit` in `bridge_contract`, there should be a check that the deposit limit being set is non-zero, or within some other reasonable bound. A deposit limit of zero would effectively inhibit all future deposits.
3. Fixed When processing `op::transfer_ownership()` check that the address of the owner being assigned is non-zero.

Recommendation: Consider adding the relevant checks.

Auditor Suggestions

S1 Needed Support for No-Ops

Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
The operation is going to be used manually by authorized users in a very low frequency basis.
```

File(s) affected: `bridge_contract.fc`

Description: Currently, an authorized address can waste gas on setting either the global deposit flag or the authorization of an address to a value it already was.

Recommendation: If the specified value does not create a change in the contract's state, it should be treated as a no-op, and the message value can be returned to the caller.

S2 Critical Role Transfer Not Following Two-Step Pattern

Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `6d57636b45211c4b2c8fea6f4ba4f62320f08146`.
The client provided the following explanation:

```
transfer_ownership has been modified. It required now a 2 step process.
  When transfer_ownership is called it sets a pending owner, but the owner still continues to be the
owner until the new owner accepts the ownership of the contract
```

```
The new owner must now call accept_ownership in order to accept the role.
That process ensures that only a valid address can be set as the owner.
```

File(s) affected: `bridge_contract.fc`

Description: The owner of the contracts can specify `transfer_ownership()` to transfer the ownership of the `bridge_contract` to a new address. If an uncontrollable address is accidentally provided as the new owner address then the contract will no longer have an active owner. This means all funds in the contract will be inaccessible along with any functions only available to the owner.

Recommendation: Consider splitting up the process of ownership transfer into two steps. For example, by including another operation `op::set_pending_owner()`.

S3 Stylistic Improvements

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `c58cfb45d1feaf367c3b1fdb972147b4836b4e67`.
The client provided the following explanation:

```
Style recommendations have been included
```

File(s) affected: `upgrade.fc`, `bridge_contract.fc`

Description: The following suggestions can improve code readability and style:

- `remove_dict_address_key_if_exists()` can rename the parameter `slice key` to `slice address`.
- The list of authorized users is sometimes referred to as an `authorized_dict` and other times referred to as an `authorized_list`. Keeping this struct consistently named can improve code readability.
- Throughout the code, errors are often thrown using the following pattern:

```
if(breaking_condition) {
    throw(error)
}
```

Consider instead, using `throw_unless()` or `throw_if`, which can define the condition inline with the error thrown.

Recommendation: Consider including these stylistic recommendations.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `560...fb6 ./contracts/bridge_contract.fc`
- `dfe...8a3 ./contracts/imports/constants.fc`
- `dee...3f0 ./contracts/imports/upgrade.fc`
- `098...252 ./contracts/imports/ton_utils.fc`
- `b15...bc2 ./contracts/imports/log.fc`
- `55f...cb3 ./contracts/imports/lib.fc`
- `a27...cb4 ./contracts/imports/jetton_utils.fc`
- `bf0...d81 ./contracts/imports/stdlib.fc`
- `bc8...f6b ./contracts/jetton/jetton-utils.fc`
- `1b9...ce7 ./contracts/jetton/discovery-params.fc`
- `659...a7a ./contracts/jetton/params.fc`
- `887...924 ./contracts/jetton/jetton-minter.fc`
- `076...e4c ./contracts/jetton/jetton-wallet.fc`
- `e0e...693 ./contracts/jetton/jetton-minter-discoverable.fc`
- `803...3e0 ./contracts/jetton/op-codes.fc`
- `cd1...711 ./contracts/jetton/jetton-minter-ICO.fc`
- `ce9...96e ./contracts/jetton/jetton-discovery.fc`

Tests

- `975...ff7 ./tests/BridgeContract.spec.ts`

Automated Analysis

N/A

Test Suite Results

All tests are passing. This test suite is thorough, covering both unhappy and happy paths within the codebase.

Update

New features such as the two-step ownership transfer process and the jetton return are included in the updated test suite.

```
> BridgeContractFunC@0.0.1 test
> jest --verbose

 PASS  tests/BridgeContract.spec.ts (8.956 s)
  BridgeContract
    ✓ should deploy (337 ms)
    upgrade
      ✓ upgrades the contract - with data (164 ms)
      ✓ upgrades contract - without data (158 ms)
      ✓ fails to upgrade - not the owner (188 ms)
      ✓ fails to upgrade - invalid upgrade data (174 ms)
    withdraw jetton
      ✓ withdraw jetton - authorized user - enough balance (187 ms)
      ✓ withdraw jetton with native- authorized user - enough jetton balance (242 ms)
      ✓ withdraw jetton with native- authorized user - not enough jetton balance (153 ms)
      ✓ failed to withdraw - not authorized (179 ms)
      ✓ failed to withdraw - not enough contract balance (235 ms)
    withdraw native
      ✓ withdraw native - authorized user - enough balance (177 ms)
      ✓ withdraw native - authorized user - contract balance below min balance (164 ms)
      ✓ failed to withdraw - not authorized (199 ms)
    deposit native
      ✓ deposit native - limit set to max value (148 ms)
```

```
    ✓ deposit native – deposit limit not set (138 ms)
    ✓ deposit native – block global deposits (179 ms)
    ✓ deposit native – deposit exceeds limit (147 ms)
    ✓ deposit native – msg_value is too low (145 ms)
  deposits Jetton
    ✓ Omits transfer_notification from unknown jetton wallets (139 ms)
    ✓ Jetton deposit below deposit limit – success – limit set (251 ms)
    ✓ Jetton deposit – limit not set – fails – jettons returned (192 ms)
    ✓ Jetton deposit over deposit limit – fails – jettons returned (193 ms)
    ✓ Jetton deposit global deposit block – fails – jettons returned (310 ms)
  deposit limits
    ✓ success to set deposit limit info (146 ms)
    ✓ failed to set deposit limit info – not authorized (144 ms)
  transfer ownership
    ✓ success to transfer ownership (230 ms)
    ✓ failed to transfer ownership – not the owner (145 ms)
  de/authorize
    ✓ success to authorize (140 ms)
    ✓ failed to authorize – not the owner (218 ms)
    ✓ success to deauthorize (156 ms)
    ✓ failed to deauthorize – not the owner (152 ms)
  global deposits block
    ✓ success to block global deposits – authorized user (197 ms)
    ✓ success to allow global deposits – authorized user (156 ms)
    ✓ failed to allow global deposits – not authorized (152 ms)
    ✓ failed to block global deposits – not authorized (142 ms)

Test Suites: 1 passed, 1 total
Tests:       35 passed, 35 total
Snapshots:   0 total
Time:        9.007 s
Ran all test suites.
```

Changelog

- 2024-09-06 - Initial report
- 2024-09-18 - Final Report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and and may not be represented as such. No third party is entitled to rely on the report in any any way, including for the purpose of making any decisions to buy or sell a product, product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or or any open source or third-party software, code, libraries, materials, or information to, to, called by, referenced by or accessible through the report, its content, or any related related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.