Quantstamp Security Assessment Certificate

March 16th 2022 — Quantstamp Verified

Stargate

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

Executive Summary

Type	Decentralized Bridge
Auditors	Souhail Mssassi, Research Engineer Jose Ignacio Orlicki, Senior Engineer Jake Bunce, Research Engineer
Timeline	2022-01-24 through 2022-03-11
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	LayerZero Whitepaper LayerZero Blog
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
stargate	ed010ab
stargate	125cf34
stargate	87ebdb0
stargate	5f0dfd2

Total Issues	**13** (3 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**2** (0 Resolved)
Low Risk Issues	**9** (2 Resolved)
Informational Risk Issues	**1** (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	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.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
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).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

We have reviewed the code, documentation, and test suite and found several issues of various severities. Overall, we consider the code to be well-written but with some lacking documentation in the form of inline comments. The test suite is very extensive but can be improved given the suggested changes from this report. We also recommend improving the coverage of LPTokenERC20 and Router (branch coverage of LPTokenERC20 is especially bad at 33%). We have outlined very few suggestions to better follow best practices, and recommend addressing all the findings to tighten the contracts for future deployments or contract updates. We recommend addressing all the **13** findings to harden the contracts for future deployments or contract updates. We recommend against deploying the code as-is.
Update: All issues were addresses or acknowledged for commit **125cf34**.
Update: commit `87ebdb` We recommend fixing QSP11 and QSP12 (These issues were acknowledged by the team).

ID	Description	Severity	Status
QSP-1	Possible cross-chain desynchronization of token balances	High	Fixed
QSP-2	Use Of `transfer()` Instead Of `safeTransfer()`	Medium	Acknowledged
QSP-3	Unrestricted Transaction Forwarding	Medium	Acknowledged
QSP-4	Centralization Risk	Low	Acknowledged
QSP-5	For Loop Over Dynamic Array	Low	Acknowledged
QSP-6	Missing Address Verification	Low	Fixed
QSP-7	Mechanism of setting the bridge and factory could lead to a broken deployment	Low	Acknowledged
QSP-8	Missing event for paused transfers	Low	Fixed
QSP-9	Unclear Incentives To Avoid Collusion	Low	Acknowledged
QSP-10	Use of `balanceOf` in fee calculations may lead to unfavorable rewarding incentives	Low	Acknowledged
QSP-11	Division Before Multiplication	Low	Acknowledged
QSP-12	Missing validation on the `_bridgeAddress`	Low	Acknowledged
QSP-13	Solidity Version And SafeMath	Informational	Acknowledged

Quantstamp Audit Breakdown

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

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

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. 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:
   i. 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.
   ii. 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, clarify, 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.

Findings

QSP-1 Possible cross-chain desynchronization of token balances

Severity: *High Risk*

Status: Fixed

File(s) affected: `Pool.sol`

Description: The `lkbRemove` values in the cross-chain swap (swap -> swapRemote) payload did not account for the `eqReward`, which might result in token balances being desynchronized when they are exchanged between two chains. In `swapRemote`, the amount transferred to the user includes the `s.eqReward`, but not deducted from the source chain's `cp.balance`. Due to the desynchronization of cross-chain balances, it is possible that the swap will fail, resulting in user funds being permanently locked.

- This issue was discovered during internal audit of LayerZero team.

Recommendation:

- Account for `s.eqReward` while calculating `s.lkbRemove`
- Account for `s.eqReward` while calculating `delaCredit`

Update: This issue is fixed by LayerZero in the commit `a9a417` (L194,L200) by considering the `s.eqReward` in their calculations.

QSP-2 Use Of `transfer()` Instead Of `safeTransfer()`

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `contracts/OmniChainFungibleToken.sol (L63)`, `contracts/OmniChainFungibleToken.sol (L104)`

Description: The ERC20 standard token implementation functions also return the transaction status as a Boolean. It's good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with require() to ensure that, when the intended ERC20 function call returns false, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer didn't.

Recommendation: Use the `safeTransfer()` function from the `safeERC20` implementation or put the transfer call inside an assert or require to verify that it returned true.

Update: The LayerZero team acknowledges this issue but is not a problem because the function `_transfer()` reverts in case of problems. Detailed comment says *"This is an internal additional call to the underlying ERC20 functionality. This is not an ERC20 call. It will revert if the checks do not pass and therefore having "safeTransfer" is not required. ie. it's calling _transfer which does not return a boolean at all"*.

QSP-3 Unrestricted Transaction Forwarding

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `Bridge.sol`, `Router.sol`

Description: Forwarding transactions as data, metaprogramming, is useful but can bring serious security hazards. Take for example the AlphaHomora hack, where any malicious smart contract code (called *spells*) were allowed access to lending and combined with rounding issues generated serious financial losses. Also, for example, standard ERC-1363 allows generic transactions to be executed by third parties, this can allow ERC-20 or NFT sent to the contract to be stolen by malicious third-parties sending self-referential transactions.

Recommendation: Consider including white-listing features, at least during a launch period, to mitigate the risk of malicious smart contracts combined with other issues. Consider including blacklisting features to avoid self-referential transactions that can allow malicious users to manipulate the protocol (ie. filter out sending reflective transactions to LayerZero endpoints on-chain).

Update: The LayerZero team acknowledges this issue but is not a problem because they already have whitelisting of the contract. The detailed comment says *"We/the dao control the addition of any external smart contracts to the system(pools), whitelisting is just as safe as controlling these calls with onlyOwner via multisig. The hack mentioned in the suggestion refers to contracts where users had the ability to arbitrarily identify their own contracts(spells). This is not the same risk profile as our implementation. ie. Whitelisting is the same as having us control what contracts can be called/wired into the system."*.

QSP-4 Centralization Risk

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/Bridge.sol (L403)`

Description: In the contract `Bridge`, the role `owner` has the authority over the following functions:

- Approve tokens to spend.
- Change the bridge address.

Any comprise to the `owner` account may allow the hacker to take advantage of this.

Recommendation: We advise the client to carefully manage the `owner` account's private key to avoid any potential risk of being hacked.

Update: The LayerZero team acknowledges this issue but is not a problem because they are introducing MultSig wallets for DAO Governance. The detailed comment was that it is *"Noted and is part of the DOA governance we are introducing with multisgs etc."*.

QSP-5 For Loop Over Dynamic Array

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/LPStaking.sol (L163)`, `contracts/Pool.sol (L376)`, `contracts/Pool.sol (L498)`

Description: When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the

amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Recommendation: Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions. Consider flagging pools that need to be updated, and only iterate over those pools that need an update.

Update: The LayerZero team acknowledges this issue but is not a problem because they are aware of the limitations. The detailed comment was that it is *"Necessary, but we are aware there can be limitations if the loop becomes too big."*.

QSP-6 Missing Address Verification

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/StartgateFeeLibraryV01.sol (L16)`, `contracts/Bridge.sol (L204)`, `contracts/Factory.sol (L35)`, `contracts/Factory.sol (L52)`, `contracts/LPStaking.sol (L104)`, `contracts/Pool.sol (L404)`

Description: Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible or tokens may be burned in perpetuity.

Recommendation: It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Update: Fixed in this [commit](https://github.com/ryanzarick/stargate/commit/125cf34dd370efba40896ba619b0e34e6ba 73e54).

QSP-7 Mechanism of setting the bridge and factory could lead to a broken deployment

Severity: *Low Risk*

Status: Acknowledged

Description: The docstrings in `setBridgeAndFactory()` indicate that these values should only be set once. A better pattern for this is to use the constructor which will allow for immutable variable assignment at deploy time. Same applies to the Bridge `setRouter()` and Factory `setRouter()`.

Exploit Scenario: 1) Contracts are deployed and operational. 2) Actor with the `onlyOwner` role calls `setBridgeAndFactory()` with the zero address for the bridge and factory contract addresses. 3) There is no ability to change these to the correct contract addresses.

Recommendation: Set these one-time variable assignments at deploy time in the constructor.

Update: The LayerZero team acknowledges this issue but is not a problem because due to circular dependencies init calls cannot be placed everywhere. Also, they consider that if zero addresses are placed, it can still be change now. The detailed comment by the LayerZero team was that *"Circular dependencies prevent us from calling all these init calls inside of all the constructors. That being said I have added the router address inside of the bridge.sol and factory.sol constructors. Also made them immutable. 😉 "3) There is no ability to change these to the correct contract addresses." This exploit scenario is false. The one time initialize block is a check if the address is 0x0, therefore if the owner sets them to 0x0, they wont be blocked on subsequent calls until they are explicitly set to something non ZERO_ADDRESS. I have added another check to makde sure none of them can be set to 0x0 anyways though."*

QSP-8 Missing event for paused transfers

Severity: *Low Risk*

Status: Fixed

File(s) affected: `OmnichainFungibleToken.sol`

Description: Transfers of this token can be paused with the `paused` bool. Emitting an event is useful for people to track this event in production.

Recommendation: Emit and event when sending tokens is paused and unpaused.

Update: Fixed in this [commit](https://github.com/ryanzarick/stargate/commit/125cf34dd370efba40896ba619b0e34e6ba 73e54).

QSP-9 Unclear Incentives To Avoid Collusion

Severity: *Low Risk*

Status: Acknowledged

Description: One of the main features of the system design is the incorporation of independents Oracle and Relayer agents that do not collude with each other. Given that, we did not find in the documentation or implementation features to incentivize the separation of concerns between Oracle and Relayer, how these stakeholders are rewarded for valid behavior or punished for inappropriate behavior. Failed or missing incentives can result in low engagement of open participants (low number of Relayers), or the possibility of collusion when the value of the assets involved in the protocol increases over time (Oracle providers such as Chainlink or its employees can be tempted to collude with Relayers).

Recommendation: Is recommended to mitigate with ordering (change internal state before external calls) or block with reentrancy guards (example `ReentrancyGuard.sol` modifiers) all potential reentrancy situations.

Update: The LayerZero team acknowledges this issue but is not a problem because applications can specify which sets of relayers/oracles they want to use. The detailed comment by the LayerZero team was that *"This is more of a layerZero concern(different audits). That being said, applications can specify which sets of relayers/oracles they trust. They can choose their own, or go with defaults."*.

QSP-10 Use of `balanceOf` in fee calculations may lead to unfavorable rewarding incentives

Severity: *Low Risk*

Status: Acknowledged

Description: In the `getFees` function located in the `StargateFeeLibraryV02` contract the `eqReward` is calculated using the balance of pool of the `tokenAddress`, the issue here is that anyone can send directly tokens to the pool without being registered to the `deltaCredit`, thus reducing the `eqReward`.

Recommendation: Consider using a mapping to store the deposited tokens and use it in the calculation of the `currentAssetSD`.

Update: LayerZero has acknowledged the issue, and though it is a minor one that is not currently creating any problems, they are planing to fix it in the future, with `FeeLibraryV3`.

QSP-11 Division Before Multiplication

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Router.sol`

Description: Integer division might truncate, hence division before multiplication can lead to loss of precision. The issue is located in the `swap` function L128.

Recommendation: We advise to perform multiplication before division to avoid loss of precision.

Update: Acknowledged by the team, knowing that it is intended to truncate some dust decimal points. Otherwise, the book will get imbalanced.

QSP-12 Missing validation on the `_bridgeAddress`

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Bridge.sol`

Description: In the `setBridge` function, it takes as a parameter the `_bridgeAddress` to update the `bridgeLookup` mapping, the issue here is that the `_bridgeAddress` can be the address 0. `(Bridge.sol[L204])`

Recommendation: Consider verifying the `_bridgeAddress` to be different than address 0.

Update: Acknowledged by the team, knowing that it is a configuration problem. As Stargate is connecting to all possible chains with arbitrary address size, there is no way to confirm the validity of the address.

QSP-13 Solidity Version And SafeMath

Severity: *Informational*

Status: Acknowledged

Description: The version of Solidity in use across the project is 0.7.6. From versions >=0.8.0 SafeMath is included in the compiler so developers do not need to include this library or remember to use it correctly.

Recommendation: Use a version of Solidity >=0.8.0.

Update: The LayerZero team acknowledges this issue but is not a problem because version 0.8.* is not gas efficient enough for them and they have all the tests needed for SafeMath. The detailed comment by the LayerZero team was that "*8+ is not as gas efficient and we have tests for all the instances of safemath we need.*".

Code Documentation

Missing more online comments in smart contracts usually adds to maintainability and audits although is common for devs to avoid or implement them when the code is more stable.

Test Results

Test Suite Results

The number of tests climbed to 227 test cases for 13 modules, around 15 test cases per module on average. We consider the testing suite very complete and extensive.

```
$ npx hardhat test

  Bridge:
     ✓ constructor() - reverts for 0x0 LZ endpoint (66ms)
     ✓ constructor() - reverts for 0x0 router endpoint (43ms)
     ✓ renounceOwnership() - does not affect ownership (43ms)
     ✓ swap() - reverts when non owner
     ✓ redeemLocalCallback() - reverts when non owner
     ✓ lzReceive() - reverts for non LZ endpoint
     ✓ lzReceive() - does NOT revert if invalid function type passed (224ms)
     ✓ lzReceive() - reverts for mismatched bridgeLookup (53ms)
     ✓ setBridge() (39ms)
     ✓ setBridge() - reverts when bridge already set  (44ms)
     ✓ setBridge() - reverts for non owner
     ✓ setGasAmount() - reverts for non owner
     ✓ setGasAmount() - reverts for invalid function type
     ✓ setGasAmount()
     ✓ approveTokenSpender() - reverts for non owner
     ✓ approveTokenSpender() - approves amount
     ✓ setUseLayerZeroToken() - reverts for non owner
     ✓ setUseLayerZeroToken()
     ✓ quoteLayerZeroFee() - TYPE_SWAP_REMOTE returns valid fee
     ✓ quoteLayerZeroFee() - TYPE_ADD_LIQUIDITY returns valid fee
     ✓ quoteLayerZeroFee() - TYPE_REDEEM_LOCAL_CALL_BACK returns valid fee
     ✓ quoteLayerZeroFee() - TYPE_WITHDRAW_REMOTE returns valid fee
     ✓ quoteLayerZeroFee() - reverts with unsupported tx type is sent
     ✓ setSendVersion()
     ✓ setReceiveVersion()
     ✓ setSendVersion() - reverts when non owner
     ✓ setReceiveVersion() - reverts when non owner
     ✓ setConfig()
     ✓ setConfig() - reverts when non owner
     ✓ forceResumeReceive()
     ✓ forceResumeReceive() - reverts when non owner

  ExecuteAction:
    - executeTransaction() - randomized tests

  Factory:
     ✓ constructor() - reverts when router is 0x0
     ✓ setDefaultFeelLibrary() - reverts when router is 0x0 (39ms)
     ✓ allPoolsLength()
     ✓ createPool() - reverts if creatPair() is called for existing _poolId (91ms)
     ✓ createPool() - increments allPoolsLength() (651ms)
     ✓ createPool() - reverts when called by non router
     ✓ renounceOwnership() doesnt affect ownership (49ms)

  LPStaking:
     ✓ constructor() - reverts for bad params (65ms)
     ✓ deposit() (246ms)
     ✓ deposit() (335ms)
     ✓ deposit() and withdraw() - changes the lp amount stored in lpBalance (116ms)
     ✓ emergencyWithdraw() - changes the lp amount stored in lpBalance (168ms)
     ✓ add() - reverts with duplicate token (53ms)
     ✓ add() - reverts with 0x0 token
     ✓ withdraw() - withdraws if amount is too large (182ms)
```

```
    ✓ withdraw() - withdraw exceeds the amount owned by the lp contract (183ms)
    ✓ renounceOwnership() - onlyOwner modifiers dont block when owner doesnt exist (53ms)
    ✓ getMultiplier() - _to field equal to bonus end block
    ✓ getMultiplier() - _from field less than the bonus end block
    ✓ getMultiplier() - _from field greater than the bonus end block
    ✓ getMultiplier() - _to is > bonusEndblock and _from is < bonusEndblock
    ✓ setStargatePerBlock() - reverts when non owner
    ✓ setStargatePerBlock() - reverts when non owner
    ✓ poolLength() - reverts when non owner
    ✓ updatePool() - lpSupply is 0 (89ms)
    ✓ updatePool() - lp staking that starts in the future (92ms)

  LPTokenERC20:
    ✓ approve() - emit event / set
    ✓ increaseApproval()
    ✓ decreaseApproval()
    ✓ permit() - reverts if deadline is before block timestamp
    ✓ permit() (910ms)
    ✓ permit() - reverts with invalid signature (107ms)

  Oft:
    ✓ send() - transfers to last chain (109ms)
    ✓ send() - transfer to next chain (96ms)
    ✓ send() - send tokens to themselves (93ms)
    ✓ send() - everyone gets tokens, then everyone sends to everyone (21684ms)
    ✓ balanceOf() - initial balances of main and children (token contract)
    ✓ balanceOf() - initial balances of main and children (users) (44ms)
    ✓ sendTokens() - tokens get moved accross chains (150ms)
    ✓ lzReceive() - reverts for non owner
    ✓ estimateSendTokensFee()
    ✓ setSendVersion()
    ✓ setReceiveVersion() (412ms)
    ✓ setSendVersion() - reverts when non owner
    ✓ setReceiveVersion() - reverts when non owner
    ✓ setConfig()
    ✓ setConfig() - reverts when non owner
    ✓ forceResumeReceive()
    ✓ forceResumeReceive() - reverts when non owner

  Pool
    ✓ constructor() - reverts for 0x0 params (98ms)
    ✓ constructor() - globals are set properly (57ms)
    ✓ createChainPath() - creates a proper pool connection (66ms)
    ✓ mint() - reverts when called by non Router
    ✓ mint() - reverts if there are no chain paths (43ms)
    ✓ mint() - reverts with no weights for chainpaths (51ms)
    ✓ mint() - mints to user (88ms)
    ✓ transferFrom() (179ms)
    ✓ transferFrom() - reverts if the transfer is not approved (170ms)
    ✓ createChainPath() - weights are set (68ms)
    ✓ getChainPathsLength() (67ms)
    ✓ setWeightForChainPath() - properly allocate to two pools based on weights (229ms)
    ✓ creditChainPath() - adds to balance for remote chain (102ms)
    ✓ amountLPtoLD() - reverts when totalSupply is 0
    ✓ getChainPath() - reverts when local chain path does not exist (53ms)
    ✓ redeemLocal()  (121ms)
    ✓ redeemLocal() - reverts if path is not activated (98ms)
    ✓ creditChainPath() - emits event (59ms)
    ✓ swapRemote() - emits event (122ms)
    ✓ withdrawMintFeeBalance() - emits event (126ms)
    ✓ withdrawMintFeeBalance() - reverts when to address is 0x0 (156ms)
    ✓ redeemLocalCallback() - emits event when _amountToMintSD is > 0 (125ms)
    ✓ redeemLocalCallback() - setup to call _delta() where total > _amountSD (252ms)
    ✓ redeemLocalCheckOnRemote() - emits event (56ms)
    ✓ createChainPath() - emit correct event
    ✓ setWeightForChainPath() - emit correct event (55ms)
    ✓ setFee() - emits correct event
    ✓ swap() - reverts when called by non Router
    ✓ swap() - reverts if swapStop called (44ms)
    ✓ swap() - reverts if chainPath not active (79ms)
    ✓ sendCredits() - reverts when called by non Router
    ✓ sendCredits() - emits event (141ms)
    ✓ redeemRemote() - reverts when _from is 0x0
    ✓ redeemLocal() - reverts when _from is 0x0
    ✓ redeemRemote() - reverts when called by non Router
    ✓ redeemRemote() - reverts when trying to burn and totalSupply is 0 (76ms)
    ✓ activateChainPath() - reverts when called on already activated path (75ms)
    ✓ createChainPath() - reverts when duplicate chainpath tried to be created (62ms)
    ✓ activateChainPath() - reverts when called on a cp that doesnt exist (45ms)
    ✓ redeemLocal() - reverts when called by non Router
    ✓ creditChainPath() - reverts when called by non Router
    ✓ swapRemote() - reverts when called by non Router
    ✓ redeemLocalCallback() - reverts when called by non Router
    ✓ redeemLocalCheckOnRemote() - reverts when called by non Router
    ✓ createChainPath() - reverts when called by non Router
    ✓ setWeightForChainPath() - reverts when called by non Router
    ✓ setWeightForChainPath() - reverts when no chainPaths have been created yet
    ✓ setFee() - reverts when called by non Router
    ✓ setFee() - reverts cumulative fee exceeds 100%
    ✓ setFeeLibrary() - sets properly
    ✓ setFeeLibrary() - reverts by non-router
    ✓ setFeeLibrary() - reverts library address is 0x0
    ✓ setSwapStop() - set / emit event
    ✓ setSwapStop() - reverts by non router
    ✓ setDeltaParam() - reverts by non-router
    ✓ setDeltaParam() - reverts if basis points are wrong
    ✓ setDeltaParam() - emits event
    ✓ callDelta() - reverts by non-router
    ✓ withdrawProtocolFeeBalance() - reverts when called by non Router
    ✓ withdrawMintFeeBalance() - reverts when called by non Router
    ✓ withdrawMintFeeBalance() - no event when mintFeeBalance equal to 0
    ✓ withdrawProtocolFeeBalance() - no event when protocolFeeBalance equal to 0
    ✓ createChainPath() - x6 and mint() which calls _distribute fees (479ms)
    ✓ createChainPath() - x10 and mint() which calls _distribute fees (842ms)
    ✓ createChainPath() - x50 and mint() which calls _distribute fees (7238ms)

  Pool State:
    ✓ swap() - lzTxParams transfers extra gas (1806ms)
    ✓ swap() - reverts with 0 amount
    ✓ swap() - reverts when cp balance is not high enough for swap (3098ms)
    ✓ swap() - reverts when cp balance is not high enough for swap (3224ms)
    ✓ swap() - using loop back mock, revert on sgReceive when paused (2292ms)
    ✓ swap() - using loop back mock, can send on sgReceive (2094ms)
    ✓ redeemLocal() - reverts when lp isnt enough (325ms)
    ✓ instantRedeemLocal() - reverts when totalLiquidity is 0 (53ms)
   LP pools are filled and fees set:
      ✓ delta() - run a series of tests NOT in fullMode and ensure audit still works (7018ms)
      ✓ redeemRemote() - add eqReward to the deltaCredits (2562ms)
      ✓ redeemRemote() - nativeGasParams blocks (190ms)
      ✓ redeemRemote() - calls delta when lpDeltaBP is 0 (586ms)
      ✓ redeemRemote() - reverts when not enough lp
      ✓ redeemRemote() - lzTxParams transfers extra gas (332ms)
      ✓ redeemLocal() (750ms)
      ✓ redeemLocal() - lzTxParams transfers extra gas (441ms)
      ✓ retryRevert() - reverts when you try to send an invalid function (262ms)
      ✓ addLiquidity() - reverts when the safeTransferFrom fails (75ms)
      ✓ redeemLocalCheckOnRemote() - stores a payload on failed msg, then clears upon pool creation (654ms)
      ✓ redeemLocalCallback() - stores a payload on failed msg (246ms)
      ✓ swapRemote() - stores a payload on failed msg (225ms)
      ✓ instantRedeemLocal() - redeems less than the cap (187ms)
      ✓ instantRedeemLocal() - only burns/redeems the cap (199ms)
      ✓ instantRedeemLocal() - reverts when from address is 0x0 (206ms)
      ✓ redeemLocalCallback() - mints to user (120ms)
      ✓ redeemLocal() - reverts when amountSD is 0 (70ms)

  Router
    ✓ setBridgeAndFactory() - reverts when bridge already initialized (771ms)
    ✓ setBridgeAndFactory() - reverts when factory already initialized (107ms)
    ✓ setBridgeAndFactory() - reverts when bridge is 0x0 (104ms)
    ✓ setBridgeAndFactory() - reverts when factory is 0x0 (100ms)
    ✓ addLiquidity() - reverts for non existant pool
    ✓ createPool() - reverts when token is 0x0
    ✓ swap() - reverts when refund address is 0x0
    ✓ redeemRemote() - reverts when refund address is 0x0
    ✓ redeemRemote() - reverts when amount LP is 0
    ✓ instantRedeemLocal() - reverts with 0 lp
    ✓ redeemLocal() - reverts when refund address is 0x0
    ✓ sendCredits() - Reverts when refund address is 0x0
    ✓ retryRevert() - reverts when theres nothing to retry
    ✓ revertRedeemLocal() - reverts when ZERO non refund address
    ✓ revertRedeemLocal() - reverts when theres nothing to try to retry
    ✓ clearCachedSwap() - reverts when nothing to clean
    ✓ creditChainPath() - reverts when caller is not Bridge
    ✓ removeLiquitidyRemote() - reverts when caller is not Bridge
    ✓ redeemLocalCallback() - reverts when caller is no Bridge
    ✓ redeemLocalCallback() - emits event (183ms)
    ✓ swapRemote() - reverts when caller is no Bridge
    ✓ createPool() - reverts with non owner
    ✓ createChainPath() - reverts with non owner
    ✓ setWeightForChainPath() - reverts when caller is not the dao
    ✓ setWeightForChainPath() (289ms)
    ✓ setProtocolFeeOwner() - reverts when caller is not the dao
    ✓ setProtocolFeeOwner() - reverts when fee owner is 0x0
    ✓ setMintFeeOwner() - reverts when non owner
    ✓ setMintFeeOwner() - reverts when fee owner is 0x0
    ✓ setFees() - reverts when caller is not the dao
```

```
    ✓ setFeeLibrary() - reverts when non owner
    ✓ setSwapStop() - reverts when non owner
    ✓ setFeeLibrary() - when caller is Owner (182ms)
    ✓ setSwapStop() - when caller is the Owner (160ms)
    ✓ callDelta() - anyone can call (144ms)
    ✓ withdrawMintFee() - reverts when non owner
    ✓ withdrawMintFee() (172ms)
    ✓ withdrawProtocolFee() - reverts when non owner
    ✓ withdrawProtocolFee() - reverts when non owner (184ms)

  StargateFeeLibraryV02:
    ✓ getEquilibriumFee() (532ms)

  StargateToke:
    ✓ name()
    ✓ symbol()
    ✓ decimals()
    ✓ constructor() - mints to deployer
    ✓ renounceOwnership() - doesnt affect ownership
    ✓ lzReceive() - can mint to an address once wired (196ms)
    ✓ setConfig() - reverts when non owner
    ✓ setDestination() - reverts with non owner
    ✓ sendTokens() (111ms)
    ✓ pauseSendTokens() - reverts with non owner
    ✓ pauseSendTokens() (53ms)
    ✓ pauseSendTokens() - cant transfer (144ms)

  SwapMath
    ✓ no fee swap test, vanilla delta (1959ms)
    ✓ swap test, with sg and lp fee, vanilla delta (2256ms)
    ✓ swap test, with all fee, vanilla delta (2249ms)
    ✓ add in a new stargateD, alice addLiquidity, no fee, then swap() (6226ms)


  227 passing (5m)
```

Code Coverage

The code coverage is very good, statement coverage is above 99%, branch coverage is 100%.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	99.83	100	99.25	99.83	
Bridge.sol	100	100	100	100	
Factory.sol	100	100	100	100	
LPStaking.sol	100	100	100	100	
LPTokenERC20.sol	100	100	100	100	
OmnichainFungibleToken.sol	96.43	100	91.67	96.55	51
Pool.sol	100	100	100	100	
Router.sol	100	100	100	100	
StargateToken.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IStargateFeeLibrary.sol	100	100	100	100	
IStargateReceiver.sol	100	100	100	100	
IStargateRouter.sol	100	100	100	100	
All files	**99.83**	**100**	**99.25**	**99.83**	

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.

Contracts

```
4ecac97441b697dc3ec17bd81f052316a6fd89bd3692ef3d8f2e83f02fc9b0ea ./contracts/LPStaking.sol
a6743207fd2c71aee625cd80ddd2aa504f110d43b975312f2bd238a7af6a2f63 ./contracts/Factory.sol
95e39c5925870fbb390c11c62b377e295d2638dd0d9c4faf96fd2e5baa694877 ./contracts/Pool.sol
1a7e6e24c696d98da8809cf95f1317e6d001e7b4800f524855e5fb66b10ff1c9 ./contracts/Bridge.sol
fcb9419aaab336c4c9efb3159e7d8af12607d0a7ff2526396c85a1df23865841 ./contracts/LPTokenERC20.sol
c102be6043e9d6d6be012f50db15a1ec051ecfabdc16386c2a0d5b36df713325 ./contracts/StargateToken.sol
71fad578f851f77dd11771bec73c838cc27c8521a3b1230b5be7a2f3eb5dedc4 ./contracts/OmnichainFungibleToken.sol
7664ea69cf5b5da21d4f90a1deff5ad9d58880f7c0c52c997dbc1ff719fb4457 ./contracts/Router.sol
52cc616773aeebaf7da6a4021370062b18151baf5c8ac8d2e162745390bfb61b ./contracts/Staking.sol
84f12afeab73920c7e22f967cf39ac462e27bb42873da9010b2917dc9beb89e4 ./contracts/interfaces/IStargateRouter.sol
1312650cb9baf8a0ab69bca273fc9e38d8e41038891957cd8ba75dfacbad41fd ./contracts/interfaces/IStargateReceiver.sol
af0c15b42c47173326e384eadc5ddda50be769b691c0399dd9ea452b1bfbbeab ./contracts/interfaces/ILayerZeroEndpoint.sol
1511ad1ebb4e2cf4e9932ff1a3bbcaa0c1aadaef560e004554a709967482beda ./contracts/interfaces/ILayerZeroReceiver.sol
2a5966def85d40f82c7da2d8219e4bdc73b2a2f6e507117297a115f02556a302 ./contracts/interfaces/ILayerZeroUserApplicationConfig.sol
9660f4443882935c96e1d7e3fe26b0b5fdf4164b281f9210fd761beddc72e45b ./contracts/interfaces/IStargateFeeLibrary.sol
a0097cfbf35c939a4dc1ad6cd9d655c4e0732d4e1936776a8c103d5469b4cafc ./contracts/libraries/StargateFeeLibraryV01.sol
6bc24afb1868dfb8b41bc969fb8b9c804893f9d52f258dbad62ad2860a97af86 ./contracts/mocks/MockTokenWithDecimals.sol
e061cc44889a8cd1e7f129a70a67beaf08098ccbabd3ad868ebbce9d8c05dfcd ./contracts/mocks/MockToken.sol
52bdb1775bd05505a89426c103e73b28aa66f94ac1ab6187aed511bbfc45bfbf ./contracts/mocks/LayerZeroEndpointMock.sol
18babd68d11db0f15aa4d02f2328a1af1dea96e2ed1466c73f7e602d52b3f0e6 ./contracts/mocks/MockTokenNoInitialMint.sol
595c56b714430ab6f991b21fa81e62663dfc3501659672c6b738f39403ee24f0 ./contracts/mocks/LayerZeroMultiEndpointMock.sol
```

Tests

```
0e18f5706efc069034e40b786d4bc76efc84ee6e931e84fbd4bd7f4c868c9a07 ./test/oft.test.js
c97225112c609b1740f4fac769ea77f48e08857b42c1cffae5d0178334de0429 ./test/PoolState.test.js
ba1355c685c2fcd22c72aad0ce5090f96dae3ceaf77467b50f224559051818e5 ./test/Factory.test.js
a460261a26bef8fff3ca2c3c97353a5621639dfe41f8bf3d21eeee9a8b336a31 ./test/Pool.test.js
560401a29cf416d13a7d4366bc4fa07ee264032dba9a71c149310fee771ffcc6 ./test/LPTokenERC20.test.js
07ec760bc1121ee92c1759402e08b9f3e2b5737160e017a6c55692539b82f171 ./test/Stargate.test.js
81cf787834ae2658c2d6bdf0f6ced6b630eac45165916404ebb5538904442560 ./test/Staking.test.js
bdfccf2c2b301dd24c89621b196082b5080e75902e41e6d10005e4be1d2131b4 ./test/LPStaking.test.js
cd084fc8e84b1e11d86d72db0943259594037250eb820dcb0861d88fa0fb3641 ./test/StargateToken.test.js
c2a50a3e9f0826bec005998b756104de98df67df21705ea3c9e3c206592846f0 ./test/Bridge.test.js
4a02a6c0c9a2cc096321c8a08595f67d8eb236df9a2966d74fc73dc5d2a32f98 ./test/SwapMath.test.js
160a1da3f6ef49c47eee66a06a3271381fa797738e5f589f40f6bbb866b1f611 ./test/Router.test.js
77956b7903da8f6603a5379562496aabf047c1ba27a2ce90f28d27fbd0349f39 ./test/util/poolstateHelpers.js
231b1ff8ad0fee6fb0c33fab701dc14f0c612d6b0fbf77fac9c67d1a1b29cd59 ./test/util/globalBook.js
019b62f0df8d5fd2bfebaef430112787220e235d2b54b7e37ef79108462cc9fc ./test/util/helpers.js
```

Changelog

- 2022-02-11 - Initial report
- 2022-02-24 - Reaudit report (125cf34)
- 2022-03-11 - Reaudit report (87ebdb)
- 2022-03-16 - Reaudit report (5f0dfd2)

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.