
# SMART CONTRACT AUDIT

ZOKYO.

March 21th, 2022 | v. 1.1

## PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

# TECHNICAL SUMMARY

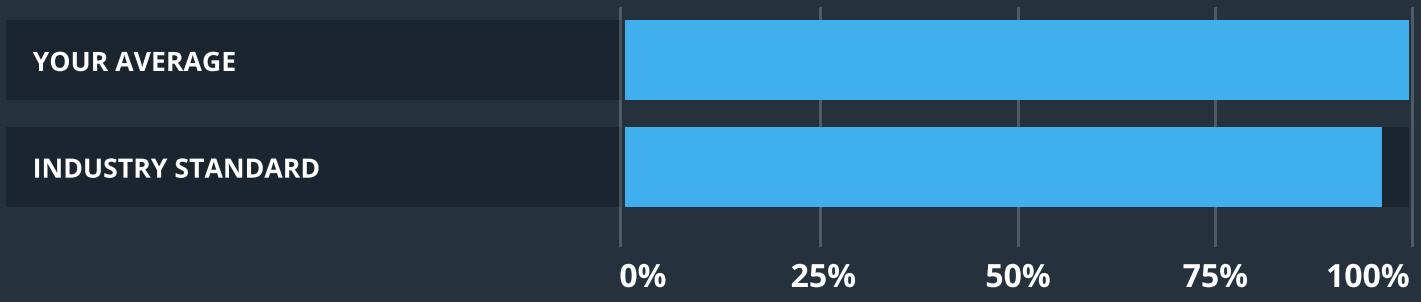
This document outlines the overall security of the LayerZero smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the LayerZero smart contract codebase for quality, security, and correctness.

## Contract Status


There were some critical, high and medium issues found during the audit.

## Testable Code




The testable code is 99%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a security of the contract we at Zokyo recommend that the LayerZero team put in place a bug bounty program to encourage further and active analysis of the smart contract.

# TABLE OF CONTENTS

# AUDITING STRATEGY AND TECHNIQUES APPLIED

The LayerZero smart contract's source code was taken from the repository provided by the LayerZero team: https://github.com/ryanzarick/stargate

Commit (audited): ec35a22b8d4be6d45144c11f733967b4d1367ac8
Latest commit (post-audit): 6fbc8821b1fa98c103338cdee383a78441182ba7
For the report version 1.1 the code was open-sourced by the LayerZero team and moved to the repository: https://github.com/stargate-protocol/stargate
Latest commit (post-audit): dbd4dda18891c996233b18e92f9f493348ea8e22
The code was verified by Zokyo team to be the same as after the initial audit with all fixes included

Within the scope of this audit Zokyo auditors have reviewed the following contract(s):

- Bridge.sol
- Factory.sol
- LPStaking.sol
- LPTokenERC20.sol
- OmnichainFungibleToken.sol
- Pool.sol
- Router.sol
- StargateToken.sol

**Throughout the review process, care was taken to ensure that the contract:**

- Implements and adheres to existing standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of resources, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of LayerZero smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

| | | | |
|---|---|---|---|
| 1 | Due diligence in assessing the overall code quality of the codebase. | 3 | Testing contract logic against common and uncommon attack vectors. |
| 2 | Cross-comparison with other, similar smart contracts by industry leaders. | 4 | Thorough, manual review of the codebase, line-by-line. |

# EXECUTIVE SUMMARY

There were several issues found during the audit, which were successfully resolved by the LayerZero team. The overall quality of the code is high, it is well documented and follows best practices. There were issues connected to the fees calculations though they were successfully resolved by the LayerZero team.

Also it needs to be mentioned, that contracts have excellent native coverage with unit tests. Nevertheless all scenarios were carefully checked by the auditor's team in order to verify the completeness and meaningfulness.

# STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

### Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

### High

The issue affects the ability of the contract to compile or operate in a significant way.

### Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

### Low

The issue has minimal impact on the contract’s ability to operate.

### Informational

The issue has no impact on the contract’s ability to operate.

# COMPLETE ANALYSIS

**Pool.sol**

MEDIUM | RESOLVED

**Subtraction might revert.**

Line 181. Storage variable ‘eqFeePool’ collects fees in function swapRemote() and subtracts fees in function swap(). However, in case swap() is executed before swapRemote() has been executed, subtraction ‘eqFeePool = eqFeePool.sub(s.eqReward);’ might revert.

**Recommendation:**
Make sure that subtraction won’t revert and prevent function call.

INFORMATIONAL | RESOLVED

**Use a storage constant.**

Lines 201, 234, 277, 403, 425, 585, 604. Value ‘10000’ is used across these lines directly and should be moved to storage constant.

**Recommendation:**
Move value ’10000’ to storage constant and use it instead.

## Router.sol

LOW | RESOLVED

### Variables lack validation.

Function setBridgeAndFactory(). Function parameters '_bridge' and '_factory' should be validated not to be zero addresses.

**Recommendation:**
Add validation that function parameters are not zero addresses.

## Factory.sol

INFORMATIONAL | RESOLVED

### Variable lacks validation.

Function setRouter(). Function parameter '_router' should be validated not to be zero address.

**Recommendation:**
Add validation that the function parameter is not zero address.

## LPStacking.sol

LOW | RESOLVED

**Variable lacks validation.**

Lines 199, 220. Variable ‘pending’ should be validated not to be zero before transferring rewards to reduce gas spendings and transfer of zero value.

**Recommendation:**
Transfer rewards only in case ‘pending’ is greater than 0.

INFORMATIONAL | RESOLVED

**Variables lack validation.**

constructor(). Parameters ‘_bonusEndBlock’ should be validated to be greater than ‘_startBlock’ and ‘startBlock’ should be validated to be greater than block.number.

**Recommendation:**
Add corresponding validations.

| | Bridge.sol | LPTokenERC20.sol | Factory.sol |
|---|---|---|---|
| Re-entrancy | Pass | Pass | Pass |
| Access Management Hierarchy | Pass | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass | Pass |
| Unexpected Ether | Pass | Pass | Pass |
| Delegatecall | Pass | Pass | Pass |
| Default Public Visibility | Pass | Pass | Pass |
| Hidden Malicious Code | Pass | Pass | Pass |
| Entropy Illusion (Lack of Randomness) | Pass | Pass | Pass |
| External Contract Referencing | Pass | Pass | Pass |
| Short Address/ Parameter Attack | Pass | Pass | Pass |
| Unchecked CALL Return Values | Pass | Pass | Pass |
| Race Conditions / Front Running | Pass | Pass | Pass |
| General Denial Of Service (DOS) | Pass | Pass | Pass |
| Uninitialized Storage Pointers | Pass | Pass | Pass |
| Floating Points and Precision | Pass | Pass | Pass |
| Tx.Origin Authentication | Pass | Pass | Pass |
| Signatures Replay | Pass | Pass | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass | Pass | Pass |

| | LPStaking.sol | StargateToken.sol | Pool.sol |
|---|---|---|---|
| Re-entrancy | Pass | Pass | Pass |
| Access Management Hierarchy | Pass | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass | Pass |
| Unexpected Ether | Pass | Pass | Pass |
| Delegatecall | Pass | Pass | Pass |
| Default Public Visibility | Pass | Pass | Pass |
| Hidden Malicious Code | Pass | Pass | Pass |
| Entropy Illusion (Lack of Randomness) | Pass | Pass | Pass |
| External Contract Referencing | Pass | Pass | Pass |
| Short Address/ Parameter Attack | Pass | Pass | Pass |
| Unchecked CALL Return Values | Pass | Pass | Pass |
| Race Conditions / Front Running | Pass | Pass | Pass |
| General Denial Of Service (DOS) | Pass | Pass | Pass |
| Uninitialized Storage Pointers | Pass | Pass | Pass |
| Floating Points and Precision | Pass | Pass | Pass |
| Tx.Origin Authentication | Pass | Pass | Pass |
| Signatures Replay | Pass | Pass | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass | Pass | Pass |

| | Router.sol | OmnichainFungibleToken.sol |
|---|---|---|
| Re-entrancy | Pass | Pass |
| Access Management Hierarchy | Pass | Pass |
| Arithmetic Over/Under Flows | Pass | Pass |
| Unexpected Ether | Pass | Pass |
| Delegatecall | Pass | Pass |
| Default Public Visibility | Pass | Pass |
| Hidden Malicious Code | Pass | Pass |
| Entropy Illusion (Lack of Randomness) | Pass | Pass |
| External Contract Referencing | Pass | Pass |
| Short Address/ Parameter Attack | Pass | Pass |
| Unchecked CALL Return Values | Pass | Pass |
| Race Conditions / Front Running | Pass | Pass |
| General Denial Of Service (DOS) | Pass | Pass |
| Uninitialized Storage Pointers | Pass | Pass |
| Floating Points and Precision | Pass | Pass |
| Tx.Origin Authentication | Pass | Pass |
| Signatures Replay | Pass | Pass |
| Pool Asset Security (backdoors in the underlying ERC-20) | Pass | Pass |

# CODE COVERAGE AND TEST RESULTS FOR ALL FILES

## Tests written by the LayerZero team

As part of our work assisting LayerZero team in verifying the correctness of their contract code, our team has checked the complete set of unit tests prepared by the LayerZero team.

It needs to be mentioned, that the original code has a significant original coverage with testing scenarios provided by the LayerZero team. All of them were also carefully checked by the auditors' team to be consistent and meaningful.

**Contract: Bridge**

- ✓ constructor() - reverts for 0x0 LZ endpoint (88ms)
- ✓ constructor() - reverts for 0x0 router endpoint
- ✓ renounceOwnership() - does not affect ownership (48ms)
- ✓ swap() - reverts when non owner (42ms)
- ✓ redeemLocalCallback() - reverts when non owner
- ✓ lzReceive() - reverts for non LZ endpoint
- ✓ lzReceive() - does NOT revert if invalid function type passed (425ms)
- ✓ lzReceive() - reverts for mismatched bridgeLookup
- ✓ setBridge()
- ✓ setBridge() - reverts when bridge already set (38ms)
- ✓ 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

**Contract:** **Factory**

✓ constructor() - reverts when router is 0x0 (45ms)
✓ setDefaultFeelLibrary() - reverts when router is 0x0
✓ allPoolsLength()
✓ createPool() - reverts if creatPair() is called for existing _poolId (72ms)
✓ createPool() - increments allPoolsLength() (551ms)
✓ createPool() - reverts when called by non router
✓ renounceOwnership() doesnt affect ownership

**Contract:** **LPStaking**

✓ constructor() - reverts for bad params (87ms)
✓ deposit() (205ms)
✓ deposit() (267ms)
✓ deposit() and withdraw() - changes the lp amount stored in lpBalance (90ms)
✓ emergencyWithdraw() - changes the lp amount stored in lpBalance (141ms)
✓ add() - reverts with duplicate token (45ms)
✓ withdraw() - withdraws if amount is too large (157ms)
✓ withdraw() - withdraw exceeds the amount owned by the lp contract (157ms)
✓ renounceOwnership() - onlyOwner modifiers dont block when owner doesnt exist (48ms)
✓ 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 (83ms)
✓ updatePool() - lp staking that starts in the future (79ms)

**Contract:** **LPTokenERC20**

✓ approve() - emit event / set
✓ increaseApproval()
✓ decreaseApproval()
✓ permit() - reverts if deadline is before block timestamp
✓ permit() (1457ms)
✓ permit() - reverts with invalid signature (322ms)

**Contract:** **Oft**

✓ send() - transfers to last chain (77ms)
✓ send() - transfer to next chain (82ms)
✓ send() - send tokens to themselves (76ms)
✓ send() - everyone gets tokens, then everyone sends to everyone (18704ms)
✓ balanceOf() - initial balances of main and children (token contract)
✓ balanceOf() - initial balances of main and children (users) (43ms)

✓ sendTokens() - tokens get moved accross chains (138ms)
✓ lzReceive() - reverts for non owner
✓ estimateSendTokensFee()
✓ setSendVersion()
✓ setReceiveVersion()
✓ setSendVersion() - reverts when non owner
✓ setReceiveVersion() - reverts when non owner
✓ setConfig()
✓ setConfig() - reverts when non owner
✓ forceResumeReceive()
✓ forceResumeReceive() - reverts when non owner

**Contract: Pool**

✓ constructor() - reverts for 0x0 params (144ms)
✓ constructor() - globals are set properly (58ms)
✓ createChainPath() - creates a proper pool connection (450ms)
✓ mint() - reverts when called by non Router
✓ mint() - reverts if there are no chain paths
✓ mint() - reverts with no weights for chainpaths (48ms)
✓ mint() - mints to user (72ms)
✓ transferFrom() (138ms)
✓ transferFrom() - reverts if the transfer is not approved (154ms)
✓ createChainPath() - weights are set (62ms)
✓ getChainPathsLength() (86ms)
✓ setWeightForChainPath() - properly allocate to two pools based on weights (178ms)
✓ creditChainPath() - adds to balance for remote chain (80ms)
✓ amountLPtoLD() - reverts when totalSupply is 0
✓ getChainPath() - reverts when local chain path does not exist (57ms)
✓ redeemLocal() (129ms)
✓ redeemLocal() - reverts if path is not activated (88ms)
✓ creditChainPath() - emits event (54ms)
✓ swapRemote() - emits event (199ms)
✓ withdrawMintFeeBalance() - emits event (116ms)
✓ withdrawMintFeeBalance() - reverts when to address is 0x0 (152ms)
✓ redeemLocalCallback() - emits event when _amountToMintSD is > 0 (122ms)
✓ redeemLocalCallback() - setup to call _delta() where total > _amountSD (225ms)
✓ redeemLocalCheckOnRemote() - emits event (49ms)
✓ createChainPath() - emit correct event
✓ setWeightForChainPath() - emit correct event (51ms)
✓ setFee() - emits correct event
✓ swap() - reverts when called by non Router
✓ swap() - reverts if swapStop called
✓ swap() - reverts if chainPath not active (59ms)

✓ sendCredits() - reverts when called by non Router
✓ sendCredits() - emits event (188ms)
✓ 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 (68ms)
✓ activateChainPath() - reverts when called on already activated path (76ms)
✓ createChainPath() - reverts when duplicate chainpath tried to be created (54ms)
✓ activateChainPath() - reverts when called on a cp that doesnt exist (43ms)
✓ 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 (442ms)
✓ createChainPath() - x10 and mint() which calls _distribute fees (886ms)
✓ createChainPath() - x50 and mint() which calls _distribute fees (6980ms)

**Pool State**

✓ swap() - lzTxParams transfers extra gas (1984ms)
✓ swap() - reverts with 0 amount (42ms)
✓ swap() - reverts when cp balance is not high enough for swap (3408ms)
✓ swap() - reverts when cp balance is not high enough for swap (3136ms)
✓ swap() - using loop back mock, revert on sgReceive when paused (3291ms)
✓ swap() - using loop back mock, can send on sgReceive (1744ms)

- ✓ redeemLocal() - reverts when lp isnt enough (302ms)
- ✓ instantRedeemLocal() - reverts when totalLiquidity is 0 (50ms)

LP pools are filled and fees set

- ✓ delta() - run a series of tests NOT in fullMode and ensure audit still works (5576ms)
- ✓ redeemRemote() - add eqReward to the deltaCredits (2289ms)
- ✓ redeemRemote() - nativeGasParams blocks (188ms)
- ✓ redeemRemote() - calls delta when lpDeltaBP is 0 (764ms)
- ✓ redeemRemote() - reverts when not enough lp
- ✓ redeemRemote() - lzTxParams transfers extra gas (279ms)
- ✓ redeemLocal() (654ms)
- ✓ redeemLocal() - lzTxParams transfers extra gas (397ms)
- ✓ retryRevert() - reverts when you try to send an invalid function (184ms)
- ✓ addLiquidity() - reverts when the safeTransferFrom fails (55ms)
- ✓ redeemLocalCheckOnRemote() - stores a payload on failed msg, then clears upon pool creation (599ms)
- ✓ redeemLocalCallback() - stores a payload on failed msg (203ms)
- ✓ swapRemote() - stores a payload on failed msg (211ms)
- ✓ instantRedeemLocal() - redeems less than the cap (146ms)
- ✓ instantRedeemLocal() - only burns/redeems the cap (174ms)
- ✓ instantRedeemLocal() - reverts when from address is 0x0 (165ms)
- ✓ redeemLocalCallback() - mints to user (111ms)
- ✓ redeemLocal() - reverts when amountSD is 0 (64ms)

**Contract: Router**

- ✓ setBridgeAndFactory() - reverts when bridge already initialized (120ms)
- ✓ setBridgeAndFactory() - reverts when factory already initialized (97ms)
- ✓ setBridgeAndFactory() - reverts when bridge is 0x0 (96ms)
- ✓ setBridgeAndFactory() - reverts when factory is 0x0 (96ms)
- ✓ 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 (183ms)
- ✓ 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 (150ms)
    ✓ 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() (263ms)
    ✓ 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 (138ms)
    ✓ setSwapStop() - when caller is the Owner (143ms)
    ✓ callDelta() - anyone can call (140ms)
    ✓ withdrawMintFee() - reverts when non owner
    ✓ withdrawMintFee() (158ms)
    ✓ withdrawProtocolFee() - reverts when non owner
    ✓ withdrawProtocolFee() - reverts when non owner (164ms)
  StargateFeeLibraryV02
    ✓ getEquilibriumFee() (436ms)
Contract: StargateToken
    ✓ name()
    ✓ symbol()
    ✓ decimals()
    ✓ constructor() - mints to deployer
    ✓ renounceOwnership() - doesnt affect ownership
    ✓ lzReceive() - can mint to an address once wired (179ms)
    ✓ setConfig() - reverts when non owner
    ✓ setDestination() - reverts with non owner
    ✓ sendTokens() (101ms)
    ✓ pauseSendTokens() - reverts with non owner
    ✓ pauseSendTokens() (45ms)
    ✓ pauseSendTokens() - cant transfer (113ms)
  SwapMath
    ✓ no fee swap test, vanilla delta (1465ms)
    ✓ swap test, with sg and lp fee, vanilla delta (1822ms)
    ✓ swap test, with all fee, vanilla delta (1807ms)
    ✓ add in a new stargateD, alice addLiquidity, no fee, then swap() (5336ms)


  227 passing (4m)
```

| FILE | % STMTS | % BRANCH | % FUNCS |
|---|---|---|---|
| Bridge.sol | 99.83 | 100.00 | 100.00 |
| Factory.sol | 100.00 | 100.00 | 100.00 |
| LPStaking.sol | 100.00 | 100.00 | 100.00 |
| LPTokenERC20.sol | 100.00 | 100.00 | 100.00 |
| OmnichainFungibleToken.sol | 96.43 | 100.00 | 96.55 |
| Pool.sol | 100.00 | 100.00 | 100.00 |
| Router.sol | 100.00 | 100.00 | 100.00 |
| StargateToken.sol | 100.00 | 100.00 | 100.00 |
| **All files** | **99.53** | **100.00** | **99.47** |

We are grateful to have been given the opportunity to work with the LayerZero team.

**The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.**

Zokyo's Security Team recommends that the LayerZero team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.