Quantstamp Security Assessment Certificate

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

January 23rd 2022 — Quantstamp Verified

Boba Network

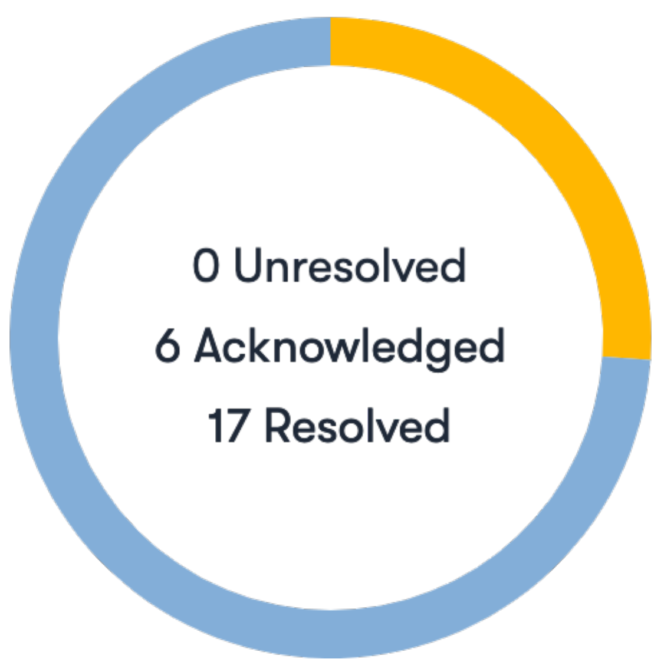
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Optimistic Rollup (Layer 2 blockchain)
Auditors	Sebastian Banescu, Senior Research Engineer Kacper Bąk, Senior Research Engineer Marius Guggenmos, Senior Research Engineer Martin Derka, Senior Research Engineer
Timeline	2021-11-15 through 2021-12-21
EVM	Altair
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Liquidity Pools
Documentation Quality	Medium
Test Quality	Medium
Diff/Fork information	The repository which was in scope for this audit is a fork of the Optimism Monorepo available at https://github.com/ethereum-optimism/optimism
Source Code	

Repository	Commit
optimism-v2 (initial audit)	426234d
optimism-v2 (reaudit)	898dfb6

Total Issues	23	(17 Resolved)
High Risk Issues	1	(1 Resolved)
Medium Risk Issues	2	(2 Resolved)
Low Risk Issues	10	(7 Resolved)
Informational Risk Issues	8	(5 Resolved)
Undetermined Risk Issues	2	(2 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

After initial audit: Quantstamp has performed an audit of the Boba optimism-v2 repository with a focus on the Boba-specific smart contracts such as the fast bridge and its associated liquidity pools on L1 and L2. During the course of this audit, Quantstamp was also asked to review the Boba Fixed Savings Contract. The audit resulted in a total of 23 findings and an additional 9 best practice violations, described below. We confirm that none of the Boba-specific tests are failing when executed on our end, but we do note that computing code coverage for these contracts was not facilitated and remains undetermined. We recommend that all issues reported in this document be addressed.

After reaudit: Quantstamp has checked the commit hash `898dfb6` and has determined that 17 issues have been resolved (that is either fixed or mitigated) and 6 issues have been acknowledged by the Boba team. More details regarding each of the issues are provided in the update messages below each issue recommendation.

ID	Description	Severity	Status
QSP-1	Missing Reconciliation After Chain Reorgs	High	Mitigated
QSP-2	Liquidity Provider Funds May Be Locked	Medium	Fixed
QSP-3	Frozen Funds	Medium	Fixed
QSP-4	Critical Updates Are Single Step	Low	Acknowledged
QSP-5	Undocumented Precision	Low	Fixed
QSP-6	Missing Input Validation	Low	Mitigated
QSP-7	Losing Gas Funds When Moving From L2 To L1	Low	Acknowledged
QSP-8	Ambiguous Checks When Adding Liquidity	Low	Fixed
QSP-9	Possible Zero Address Owner for `L1LiquidityPool`	Low	Fixed
QSP-10	Functions Callable Before Initialization	Low	Fixed
QSP-11	Violation of Checks-Effects-Interactions Pattern	Low	Fixed
QSP-12	Lacking Precision In Rewards Calculation	Low	Acknowledged
QSP-13	Pools Registered Only On L1 May Lead To Loss Of Funds	Low	Fixed
QSP-14	`rewardDebt` Variable In Liquidity Pools May Be Simplified	Informational	Acknowledged
QSP-15	Outdated Comments	Informational	Fixed
QSP-16	Insufficient Documentation For `replyNeeded`	Informational	Fixed
QSP-17	Unlocked Pragma	Informational	Acknowledged
QSP-18	Privileged Roles and Ownership	Informational	Acknowledged
QSP-19	Transaction Order Dependence Between `close()` And `expire()`	Informational	Mitigated
QSP-20	`stopStakingContract()` May Be Called Multiple Times	Informational	Fixed
QSP-21	Insufficient Events Emitted	Informational	Mitigated
QSP-22	Functions Missing `onlyInitialized` Modifier	? Undetermined	Fixed
QSP-23	Potential Gas Cost Manipulation	? Undetermined	Mitigated

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.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.1

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Missing Reconciliation After Chain Reorgs

Severity: *High Risk*

Status: Mitigated

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: This vulnerability is based on the assumption that a short-lived 51% attack on the Ethereum mainnet is feasible, and targets the liquidity providers of the Boba fast bridge with double-spending. This is a valid assumption as 51% on mainnet should cost about $2M/hour (referencing https://www.crypto51.app/, possibly with 80% discount). We briefly researched the validity of this claim about cost, and it may be mildly outdated, however the concept of the attack remains valid and the attack can be executed in the presence of bridges from other chains with weaker security guarantees than mainnet, and the attack can still damage the liquidity held in Boba's mainnet bridge. Boba's bridge contract currently holds about $3M in assets as per Etherscan's calculation, (see the value is taken at block 13809310).

Exploit Scenario: The attack works as follows. Mallory starts a 51% attack on mainnet. Then she takes some substantial liquidity and bridges it to Boba, using either the regular or fast bridge, within transaction `tx1`. This in practice takes 5 - 10 minutes. As soon as she receives the liquidity on Boba, she bridges it back to mainnet using the fast bridge feature. This will result in a cross-domain entity calling the method `clientPayL1` on the aforementioned contract (it is a proxy, the implementation source code is deployed on https://etherscan.io/address/0x9dadbc52d12e24b7d7de68477c8478aed4468da6#code). This will transfer funds into Mallory's account on mainnet in transaction `tx2`. This in practice takes around 20 minutes. Finally, Mallory cancels her 51% attack and invalidates the transaction `tx1`. By invalidation, we mean that Mallory replaces the chain with a longer chain that includes all transactions except for `tx1`, which she produced using the 51% mining power. The result is the state where Mallory never spent her original funds in `tx1`, but received funds from Boba's fast bridge in `tx2`. The attack should take roughly 30 minutes, with the cost between $200,000 and $1,000,000 (as per the referenced source) on mainnet.

Recommendation: The vulnerability is enabled by the fact that the `clientPayL1` method is completely decoupled from the deposit methods: the parameters only specify how many tokens should be transferred to whom. A possible mitigation strategy would need to account for L1 chain reorgs that would affect the deposits or transfers from L1 to L2.

Update: The dev team has done the following to mitigate this issue:

> A running `depositHash` was added for both the Standard and Fast bridge deposits/swap-ons, along with having a provision for the relayer to specify the `lastHash` while relaying a transaction. To enable this, the `relayMessage` method on the `CrossDomainMessengerFast` was also made exclusive to a permissioned actor.

Quantstamp only acknowledges this to be a valid mitigation under the assumptions that:

- The Boba Network actively detects a re-org on mainnet (so that if anyone has ETH bridged from mainnet on Boba, and the bridging transaction and the respective hash disappears, the ETH will disappear on Boba)
- The relayer bridging ETH from Boba is trusted and honest (so it always executes the method with the latest hash, which will make the release of ETH on mainnet fail if the latest hash disappeared)
- There is proper synchronization between the mainnet re-org resolution and the trusted relayer acting (goes in hand with the first point - if ETH should not be present on Boba, the relayer will know that some mitigation is in progress and will not release ETH on mainnet until the resolution is finished and the validity of the exit is confirmed)
- The hash on mainnet takes into account the amounts that are being bridged to Boba (so that instead of rolling the entire transfer back via 51% attack, I cannot drop the number from 1000 ETH to 0.1 ETH and receive the same bridging hash)
- Everything is implemented correctly.

Quantstamp has not checked these assumptions as some elements are out of scope for this audit (particularly the off-chain elements) and has asked the Boba team to confirm these assumptions. The Boba team has indicated that they believe the assumptions hold partially, not fully.

QSP-2 Liquidity Provider Funds May Be Locked

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: The `rebalanceLP()` function on L1 allows the owner to move liquidity that is present in the `L1LiquidityPool` contract to L2. The funds are moved from `L1LiquidityPool` to `L1StandardBridge` and the `withdrawLiquidity()` function that an L1 liquidity provider would use to remove liquidity assumes that the funds are with `L1LiquidityPool`. There is currently no equivalent on the L2 side, which means any rebalancing is currently permanent and cannot be brought back on L1.

If the L1 liquidity pool for the fast bridge is empty, then any attempts from L1 liquidity providers to withdraw their funds will result in failed transactions since the `L1LiquidityPool` contract will not have sufficient funds. Therefore, there is no guarantee for L1 liquidity providers regarding when they would be able to withdraw their funds from the L1 pool.

A similar issue also happens on L2 in case there are too many users who transfer funds from L1 to L2 via the fast bridge, then the `L2LiquidityPool` could be depleted and L2 liquidity providers do not have any guarantees as to when they can withdraw their funds from the `L2LiquidityPool`.

Exploit Scenario: This issue can be exemplified using the following steps:

1. Alice deposits 100 ETH in the `L1LiquidityPool` as a liquidity provider by calling `addLiquidity(100 ETH)` and the L1 pool balance is now 100 ETH
2. Bob deposits 100 ETH in the `L1LiquidityPool` as a liquidity provider by calling `addLiquidity(100 ETH)` and the L1 pool balance is now 200 ETH
3. The `L1LiquidityPool` owner calls `rebalanceLP(100 ETH)` and the L1 pool balance is now 100 ETH
4. Alice withdraws her liquidity by calling `withdrawLiquidity(100 ETH)` and the L1 pool balance is now 0 ETH.

At this point Bob cannot withdraw his ETH in any way. Bob is forced to wait for either:

- new liquidity providers who deposit at least 100 ETH in the `L1LiquidityPool`
- new end-users who want to use the fast bridge from L1 to L2 for a total of at least 100 ETH.

Recommendation: There should be a guarantee that liquidity providers can withdraw their funds within a certain period of time from when they request the withdraw. If this is not possible, consider implementing a mechanism for this purpose and/or try to provide incentives so users will want to rebalance naturally. For instance, implement an equivalent operation as `rebalanceLP` from L2 to L1. Additionally, this risk should be documented clearly when providing liquidity.

Update: A `rebalanceLP` method was added to the `L2LiquidityPool` as well to enable L2 to L1 rebalancing.

QSP-3 Frozen Funds

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/BobaFixedSavings.sol`

Description: The `BobaFixedSavings.stake()` function does not check if the `stakingCloseTimestamp` value is different from `0`. This means that even after the `stopStakingContract()` function is called, the `stake()` function will continue to work. However, the `unstake()` function will no longer work due to an underflow on L113: `uint256 noOfPeriods = ((finalTimestamp - stakeData.depositTimestamp)/(LOCK_TIME + UNSTAKE_TIME)) + 1;`.

Recommendation: Add a `require()` statement to ensure that: `stakingCloseTimestamp == 0` inside of `BobaFixedSavings.stake()`.

Update: A `require` statement to ensure `stakingCloseTimestamp == 0` was added to `BobaFixedSavings`.

QSP-4 Critical Updates Are Single Step

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: Critical updates such as changing the owner of the contract can be executed in a single step, potentially leaving the contract unrecoverable if a mistake has been made in the transaction.

Recommendation: Consider implementing a two-step process where the current owner proposes a new owner and the new owner has to claim the ownership in a separate transaction. This makes sure ownership is not accidentally transferred to an inactive account.

Update: The Boba team acknolwedged this issue by saying:

> Considering the existing contracts deployed, a new var - `pendingOwner` was not added as of now to keep the process single step - the contract is under a proxy pattern which could enable updates in the extreme case of mistaken ownership transfer.

QSP-5 Undocumented Precision

Severity: Low Risk

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: The `userRewardFeeRate` and `ownerRewardFeeRate` variables implicitly use `10**3` as their denominator but this is never explicitly documented in the code. This could lead to miscalculations in case (new) developers are not aware or forget the exact precision and assume a different precision.

Recommendation: Document these assumptions and consider using a named constant in case the precision ever changes.

Update: Documented the assumptions on contracts.

QSP-6 Missing Input Validation

Severity: Low Risk

Status: Mitigated

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`, `packages/contracts/contracts/standards/L2StandardERC20.sol`, `packages/boba/contracts/contracts/libraries/Lib_ResolvedDelegateProxy.sol`, `packages/contracts/contracts/L1/messaging/L1StandardBridge.sol`, `packages/contracts/contracts/L1/messaging/L1CrossDomainMessenger.sol`, `packages/boba/contracts/contracts/L1CrossDomainMessengerFast.sol`, `packages/boba/contracts/contracts/bridges/L1NFTBridge.sol`, `packages/boba/contracts/contracts/bridges/L2NFTBridge.sol`, `packages/boba/contracts/contracts/standards/L2StandardERC721.sol`, `packages/boba/contracts/contracts/standards/L2GovernanceERC20.sol`, `packages/contracts/contracts/L2/messaging/L2CrossDomainMessenger.sol`, `packages/contracts/contracts/L2/messaging/L2StandardBridge.sol`, `packages/boba/contracts/contracts/BobaFixedSavings.sol`

Description: All functions should perform input validation of their input parameters, especially if those functions may be called by (malicious) end-users. We have identified the following instances where functions do not properly validate their input parameters:

1. The `registerPool()` function does not check if `_l2TokenAddress` is different from `address(0)`. However, throughout the code, there is an assumption that `l2TokenAddress` is different from `address(0)` if a pool is registered, which is not enforced in case the `l1TokenAddress` is different from `address(0)`.
2. The following functions have similar issues:
    - `L2StandardERC20.constructor()`
    - `Lib_ResolvedDelegateProxy.constructor()`
    - `L2LiquidityPool.initialize()`
    - `L1StandardBridge.initialize()`
    - `L1CrossDomainMessenger.initialize()`
    - `L1CrossDomainMessengerFast.initialize()`
    - `L1NFTBridge.initialize()`
    - `L2NFTBridge.initialize()`
    - `L2StandardERC721.constructor()`
    - `L2GovernanceERC20.constructor()`
    - `L2CrossDomainMessenger.constructor()`
    - `L2StandardBridge.constructor()`
    - `BobaFixedSavings.initialize()`

Recommendation:

1. Check if `_l2TokenAddress` is different from `address(0)` inside the `registerPool()` function.
2. Add relevant checks inside the list of functions specified in the description.

Update: This issue was partially fixed. Input validation was added wherever possible, except for:

1. Upstream contracts to maintain consistency,
2. L2 token contract to maintain consistency with future such deployments, which also can be deployed in any numbers for the use case.

QSP-7 Losing Gas Funds When Moving From L2 To L1

Severity: Low Risk

Status: Acknowledged

File(s) affected: `packages/boba/gas-price-oracle/README.md`

Description: Exits from L2 to L1 require paying gas to submit a transaction on L1. When users want to exit their funds, they get a gas fee estimate. The estimate is based, however, on the previous interval. Users may be able to take advantage and submit their transactions below the required fee if gas prices go up beyond the estimate.

Recommendation: Perform careful analysis to estimate how much buffer is needed to offset the gas cost.

Update: The Boba team acknowledged this issue by saying:

The extraGas estimate amount is based on the average cost of the last 100 exits, depending on the operating expenses we could adjust the amount of offset for the estimate.

QSP-8 Ambiguous Checks When Adding Liquidity

Severity: Low Risk

Status: Fixed

File(s) affected: packages/boba/contracts/contracts/LP/L1LiquidityPool.sol

Description: If an end-user wants to add an ERC20 token amount as liquidity by calling L1LiquidityPool.addLiquidity(), but accidentally also sends any amount of ETH along with this call, then the amount of ETH is deposited and the ERC20 token amount is ignored. This may happen to beginners who are not familiar with certain wallet interfaces.

This happens due to the require statement on L331: `require(msg.value != 0 || _tokenAddress != address(0), "Amount Incorrect");`, which uses a logical-OR instead of a logical-XOR to combine the 2 Boolean conditions.

It is important to note that if `msg.value == 0` and `_tokenAddress != address(0)`, then the _amount value should be required to be greater than 0. This check is missing and should be explicitly added to avoid any ambiguity.

Recommendation: It is recommended to replace the logical-OR in the aforementioned require statement with a logical-XOR. However, since Solidity does not have a native operator you need to implement a logical XOR formula using AND and OR. The error message should also be updated to say: "Either amount incorrect or token address incorrect".

Update: Another require statement was added to form logical XOR in combination with the previous require to reduce user-side errors.

QSP-9 Possible Zero Address Owner for L1LiquidityPool

Severity: Low Risk

Status: Fixed

File(s) affected: packages/boba/contracts/contracts/LP/L1LiquidityPool.sol

Description: The L1LiquidityPool.transferOwnership() function allows setting the address of the owner to the zero address because there is no check on the _newOwner input parameter similar to that in the L2LiquidityPool. This is problematic because the onlyOwner() modifier would always pass in case the `owner == address(0)`, which is undesirable due to anyone being able to call functions such as: ownerRecoverFee(), rebalanceLP(), pause() and unpause().

Recommendation: Add a check to transferOwnership() that prevents the owner being set to address(0).

Update: A validation check was added to transferOwnership.

QSP-10 Functions Callable Before Initialization

Severity: Low Risk

Status: Fixed

File(s) affected: packages/boba/contracts/contracts/BobaFixedSavings.sol

Description: There are several functions that may be called before the contract initialize() function is called, namely:

- transferOwnership()
- stopStakingContract()

Recommendation: Remove the custom condition from the onlyOwner() modifier to disallow `owner == address(0)` from being considered as an owner. Also remove the onlyOwner modifier from the initialize() function declaration.

Update: The custom condition was removed from onlyOwner on BobaFixedSavings.

QSP-11 Violation of Checks-Effects-Interactions Pattern

Severity: Low Risk

Status: Fixed

File(s) affected: packages/contracts/contracts/L1/rollup/CanonicalTransactionChain.sol, contracts/L1/rollup/CanonicalTransactionChain.sol, contracts/L1/messaging/L1CrossDomainMessenger.sol, contracts/L2/messaging/L2CrossDomainMessenger.sol, packages/boba/contracts/contracts/LP/L1LiquidityPool.sol, packages/boba/contracts/contracts/LP/L2LiquidityPool.sol

Description: The Checks-Effects-Interactions coding pattern is meant to mitigate any chance of other contracts manipulating the state of the blockchain in unexpected and possibly malicious ways before control is returned to the original contract. As the name implied, only after checking whether appropriate conditions are met and acting internally on those conditions should any external calls to, or interactions with, other contracts be done. The Checks-Effects-Interactions pattern is violated inside the following functions:

1. L1LiquidityPool.addLiquidity() where state variables such as pool.userDepositAmount and user.amount are witten after the call to an external contract, i.e. IERC20(_tokenAddress).safeTransferFrom().
2. L1LiquidityPool.clientDepositL1() where an event is emitted after direct (IERC20(_tokenAddress).safeTransferFrom()) and indirect external contract calls.
3. L1LiquidityPool.clientPayL1() where an event is emitted after external contract calls including external calls sending ETH.
4. L1LiquidityPool.clientPayL1Settlement() where an event is emitted after external contract calls including external calls sending ETH.
5. L1LiquidityPool.ownerRecoverFee() where an event is emitted after external contract calls including external calls sending ETH.
6. L1LiquidityPool.rebalanceLP() where an event is emitted after external contract calls including external calls sending ETH.
7. L1LiquidityPool.withdrawLiquidity() where an event is emitted after external contract calls including external calls sending ETH.
8. L1LiquidityPool.withdrawReward() where an event is emitted after external contract calls including external calls sending ETH.
9. L2LiquidityPool.addLiquidity() where an event is emitted after external contract calls.
10. L2LiquidityPool.clientDepositL2() where an event is emitted after external contract calls.

11. `L2LiquidityPool.clientPayL2()` where an event is emitted after external contract calls including external calls sending ETH.
12. `L2LiquidityPool.clientPayL2Settlement()` where an event is emitted after external contract calls including external calls sending ETH.
13. `L2LiquidityPool.ownerRecoverFee()` where an event is emitted after external contract calls including external calls sending ETH.
14. `L2LiquidityPool.withdrawLiquidity()` where an event is emitted after external contract calls including external calls sending ETH.
15. `L2LiquidityPool.withdrawReward()` where an event is emitted after external contract calls including external calls sending ETH.

Recommendation: Always call functions from external contracts after all state variables have been assigned and all events emitted.

Update: Events were emitted before external calls to methods wherever possible to comply with the checks-effects-interaction pattern.

QSP-12 Lacking Precision In Rewards Calculation

Severity: Low Risk

Status: Acknowledged

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`

Description: The reward calculation is based on increasing a global value that denotes the amount of rewards per token deposited in the pool. The formula for this is basically

```
rewardsPerShare += rewardDelta * SHARE_PRECISION / totalDeposits
```

`SHARE_PRECISION` is `1e12` in the current implementation, which will very likely lead to rounding errors and fewer rewards paid out than intended, locking them in the contract.

Recommendation: Increase the share precision to minimize the rounding error. For accurate results, the share precision should have as many decimals as the expected total deposits.

Update: The Boba team has acknowledged this issue by saying:

> In order to avoid chances of inaccuracy with modifying precision in existing deployed contracts, the precision has been unchanged. With the current numbers, the improvements with increased precision looks minimal, however we can address these improvements in subsequent versions of the network.

QSP-13 Pools Registered Only On L1 May Lead To Loss Of Funds

Severity: Low Risk

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`

Description: In case a pool is not registered on L2, but only on L1, a settlement payment will be triggered with token address `address(0)`, which means the refund will be paid out in ether. Since most tokens are less valuable than ether, this can lead to an attacker stealing funds from the contract.

Recommendation: Check that the pool is registered in `clientPayL2`. To properly issue a refund, the token address on the originating pool's side might have to be passed across layers.

Update: A token registration check was added to the `ClientPayL2` method. In case a `relayMessage` fails due to the token not being registered on the pool. The message can be relayed manually again after a registration takes place.

QSP-14 `rewardDebt` Variable In Liquidity Pools May Be Simplified

Severity: Informational

Status: Acknowledged

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: The usage of the variable `rewardDebt` is hard to understand, especially due to the misleading name (there is no debt involved). The variable is only used to keep track of how many rewards have been claimed already and the computation can be simplified.

Recommendation: Simplify the computation by replacing the `rewardDebt` variable with one named `lastAccUserRewardPerShare` that is updated each time the current `rewardDebt` is. Replace all instances of `user.rewardDebt = ...` with `user.lastAccUserRewardPerShare = pool.accUserRewardPerShare`.
The pending reward computation can then be simplified to the following

```
// use SafeMath methods for the actual implementation
user.pendingReward += user.amount
    * (pool.accUserRewardPerShare - user.lastAccUserRewardPerShare)
    / 1e12
```

This change should result in improved readability and a slight decrease in contract size and gas used.

Update: The Boba team has acknowledged this issue by saying:

> With the current deployments, devs/users might be familiar with the term `rewardDebt`, however this is a great suggestion, and can be simplified in the subsequent versions of the network.

QSP-15 Outdated Comments

Severity: Informational

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: There are several lines of old code that are commented out and no longer relevant in the affected contracts. Examples include but are not limited to the `clientPayL1` and `clientPayL1Settlement` functions, as well as an outdated comment present that looks like an artifact from older code.

```
// Construct calldata for L1LiquidityPool.depositToFinalize(_to, receivedAmount)
bytes memory data = abi.encodeWithSelector(
    iL2LiquidityPool.clientPayL2.selector,
    msg.sender,
    _amount,
    pool.l2TokenAddress
);
```

Recommendation: Update/remove the comments so that they are aligned with the current implementation.

Update: Comments were updated, outdated comments removed.

QSP-16 Insufficient Documentation For `replyNeeded`

Severity: Informational

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`

Description: `clientPayL1` and `clientPayL2` make use of a function local variable called `replyNeeded`. It is a boolean variable that is used to store whether a message needs to be sent to the other layer due to not enough funds being available for the requested transfer. The current name does not reflect that and there are no comments explaining what it is for.

Recommendation: Rename the variable to something more meaningful like `hasInsufficientFunds` and provide some inline comments that explain that a message needs to be sent to the other layer to revert the deposit.

Update: Added inline comments for `replyNeeded`, the variable hasn't been renamed to maintain familiarity with existing devs/users.

QSP-17 Unlocked Pragma

Severity: Informational

Status: Acknowledged

File(s) affected: `All contracts`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.*.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version. Alternatively, possible mitigation of this issue is to at least fix the middle number to the latest version, e.g. 0.8, which ensures that the latest version of the compiler will be used.

Update: The Boba team has acknowledged this issue by saying:

> To avoid inconsistencies with deployed contracts, pragma has been still kept unlocked, this improvement can be addressed in subsequent versions of the network.

QSP-18 Privileged Roles and Ownership

Severity: Informational

Status: Acknowledged

File(s) affected: `packages/boba/contracts/contracts/BobaFixedSavings.sol`, `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol`, `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`, `packages/contracts/contracts/libraries/resolver/Lib_AddressManager.sol`, `packages/contracts/contracts/L1/messaging/L1CrossDomainMessenger.sol`, `packages/boba/contracts/contracts/L1CrossDomainMessengerFast.sol`, `packages/boba/contracts/contracts/ERC721Genesis.sol`, `packages/boba/contracts/contracts/TokenPool.sol`, `packages/contracts/contracts/chugsplash/L1ChugSplashProxy.sol`, `packages/contracts/contracts/L2/predeploys/OVM_DeployerWhitelist.sol`, `packages/contracts/contracts/L2/predeploys/OVM_GasPriceOracle.sol`, `packages/boba/contracts/contracts/DAO/governance/Timelock.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. In the `BobaFixedSavings` smart contract, only the owner can pause, unpause and stop the staking contract. The other affected contracts offer similar capabilities to the `owner` address.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update: The Boba team has indicated that:

> Privileged roles and ownership of contracts including Dao permissions to be made clear through docs/website.

QSP-19 Transaction Order Dependence Between `close()` And `expire()`

Severity: Informational

Status: Mitigated

File(s) affected: `packages/boba/contracts/contracts/AtomicSwap.sol`

Description: There is a TOD between the functions `close()` and `expire()`, i.e., the opening user may submit `expire()` whereas the closing user may submit `close()` in the same block. The outcome depends on the ordering of the transactions.

Recommendation: Inform users that this TOD could affect the outcome of these function calls.

Update: Documented the assumptions on contracts, however it is not yet documented in end-user facing documentation.

QSP-20 `stopStakingContract()` May Be Called Multiple Times

Severity: Informational

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/BobaFixedSavings.sol`

Description: It is probably intended that the `stopStakingContract()` be callable only once. Otherwise, the amounts unstaked would be unfairly higher for those who unstaked after the `stopStakingContract()` was called later.

Recommendation: Add a `require()` statement that ensures `stakingCloseTimestamp != 0` at the beginning of `stopStakingContract()`.

Update: A `require` statement to ensure `stakingCloseTimestamp == 0` was added at the beginning of the method `stopStakingContract`.

QSP-21 Insufficient Events Emitted

Severity: *Informational*

Status: Mitigated

File(s) affected: `packages/boba/contracts/contracts/BobaFixedSavings.sol`

Description: No events are emitted by the staking and unstaking functions. Normally all state-changing functions should emit an event indicating some information about the state change. This facilitates proper monitoring of the smart contract and easy integration. Examples include, but are not limited to:

1. `L1LiquidityPool.transferOwnership()` should emit an event if the `owner` is changed successfully.
2. `L1LiquidityPool.initialize()` should emit an event after a successful initialization.
3. `L1LiquidityPool.configureGas()` should emit an event if the new gas fee and stipend are set.
4. `L2LiquidityPool.transferOwnership()` should emit an event if the `owner` is changed successfully.
5. `L2LiquidityPool.initialize()` should emit an event after a successful initialization.
6. `L2LiquidityPool.transferDAORole()` should emit an event after the pool address, `owner`, and `DAO` addresses were set.
7. `L2LiquidityPool.configureExtraGasRelay()` should emit an event if the extra gas is set.
8. `L2LiquidityPool.configureGas()` should emit an event if the gas fee is configured.

Recommendation: Declare and emit events for all state-changing functions.

Update: This issue has been partially fixed by adding more events to the indicated methods, but not to all of them.

QSP-22 Functions Missing `onlyInitialized` Modifier

Severity: *Undetermined*

Status: Fixed

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`

Description: `clientPayL1` and `clientPayL1Settlement` do not have the `onlyInitialized` modifier like their L2 equivalents. It is unclear if this was implemented intentionally or not.

Recommendation: Clarify why the `onlyInitialized` modifier is not needed in this contract, otherwise, add it to to the functions.

Update: The Boba team has provided the following explanation as to why `onlyInitialized` is not needed:

> `ClientPayL1` and `ClientPayL1Settlement` can work without the `onlyInitialized` modifier - since they cannot be called before the contract is initialised with the `relayerMessenger` address. This is due to the modifier `onlyFromCrossDomainAccount`, which checks if the call comes from a `relayerMessenger`, and before initialisation it is set to `address(0)`.

QSP-23 Potential Gas Cost Manipulation

Severity: *Undetermined*

Status: Mitigated

File(s) affected: `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol`

Description: Gas costs on Boba L2 are computed by observing the balances of the different nodes and attempting to adjust the gas cost to reflect the actual ETH spent. According to the docs, this adjustment happens gradually and has limits on how fast it can change. By crafting transactions that will cost significantly more than a typical transaction, a malicious actor might be able to drive up the gas cost for all users while hemorrhaging ether from the boba nodes.

An example of an expensive transaction is depositing some amount on L2 that should then become available on L1 but does not have enough liquidity on L1 to be paid out. This will:

1. lock the amount on the L2 side, trigger a message from L2 to L1
2. on L1 the contract will notice there is not enough liquidity and will have to send a message from L1 to L2
3. on L2 the user will be credited with the amount they attempted to withdraw.

Since the gas price is adjusted gradually, it might be possible to push up the gas price significantly without the attacker having to spend a lot of ether. While there is a mechanism that tries to discourage such an attack by having to pay a fee on the settlement, it does not prevent an attacker from using very low amounts in an empty pool due to the fee being percentage-based.

Recommendation: Clarify if this attack has a mitigation in place and if not, add a mitigation.

Update: The Boba team has indicated that this issue is mitigated by:

> The calculation of our gas price is mostly weighted around the L2 fee, making the effect of the L1 sec fee lesser in the total amount. The L2 part of the fee is fixed and adjusted by us based on our revenue.

Automated Analyses

Slither

Slither reported 342 results on the Boba-specific contracts. We filtered out the false positives and included the rest of the findings in the report.

Adherence to Specification

The Boba technical specs were extremely limited. Therefore, we have mainly relied on documentation related to Optimism for this audit. We recommend improving the existing technical specification to facilitate the maintainability and auditability of the code.

Code Documentation

1. There are some grammatically wrong revert messages present in the following contracts:

   . `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol` replace "Register" with "Registered"

```
576: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
```

   . `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol` replace "Register" with "Registered"

```
342: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
397: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
444: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
492: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
529: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
572: require(L2LiquidityPoolAddress != address(0), "L2 Liquidity Pool Not Register");
573: require(pool.l2TokenAddress != address(0), "Token Address Not Register");
```

Adherence to Best Practices

1. Hardhat tests should make use of typechain to have a fully typed contract deployment and interaction, which would allow for easier and faster test development.
2. Event parameters of type `address` should be `indexed` in order to faciliate event filtering. None of the events in `L1LiquidityPool.sol` are `indexed`.
3. Several contracts are importing Solidity files from the `@openzeppelin/contracts-upgradeable` library, however, this library is not in the list of dependencies in `package.json`.
4. Magic numbers should be replaced by named constants. There are several magic numbers in `L1LiquidityPool` and `L2LiquidityPool`, which are not clearly documented, e.g. `_configureFee(35, 15);` and `configureGas(1400000, 2300);` in the `initialize()` function.
5. There exist automatic Github Workflow scripts that are used to run static analyzers such as Slither. However, the script is not updated to include the `packages/boba/contracts` subfolder.
6. There are several skipped tests and TODOs in the test files. TODOs should be properly addressed before releasing code in production.
7. Unused imports should be removed. For example `Lib_RLPReader` in `packages/contracts/contracts/libraries/bridge/Lib_CrossDomainUtils.sol`
8. Code clones present in `packages/boba/contracts/contracts/LP/L2LiquidityPool.sol` and `packages/boba/contracts/contracts/LP/L1LiquidityPool.sol` should be avoided. Code reuse is preferred.
9. Dead code should be removed. The following instances were identified:

   . In `packages/boba/contracts/contracts/L1CrossDomainMessengerFast.sol`, the funciton `replayMessage()` calls `_sendXDomainMessage()` which always reverts. It does not look like any other contract inherits from `L1CrossDomainMessengerFast`.

   . In `packages/boba/contracts/contracts/TokenPool.sol`.

Test Results

Test Suite Results

The Boba project has an integration test suite comprising of 126 tests. When running the tests on our end we confirm that 122 tests pass and 4 tests remain pending. Of the pending tests, 2 tests are skipped explicitly with the following code comment: "SKIP: until we decide what should be done in this case". The remaining 2 tests are skipped because we do not have the mnemonic of the pool contracts deployer to be able to play the role of the owner.

```
  NFT Test

    ✓ should have a name (89676 gas)
    ✓ should generate a new ERC721 and transfer it from Bob (a1a) to Alice (a2a) (736731 gas)
    ✓ should derive an NFT Factory from a genesis NFT (2826784 gas)
    ✓ should register the NFTs address in users wallet (426219 gas)

  Basic L1<>L2 Communication
    L2 => L1
      ✓ should be able to perform a withdrawal from L2 -> L1 (91020 gas)
    L1 => L2
      ✓ should deposit from L1 -> L2 (181000 gas)
      ✓ should have a receipt with a status of 1 for a successful message (181012 gas)
     - should have a receipt with a status of 0 for a failed message

  System setup
    ✓ should use the recently deployed ERC20 TEST token and send some from L1 to L2 (286511 gas)
    ✓ should transfer ERC20 TEST token to Kate (390850 gas)

  Fee Payment Integration Tests
    ✓ should return eth_gasPrice equal to OVM_GasPriceOracle.gasPrice (83258 gas)
    ✓ Paying a nonzero but acceptable gasPrice fee (165402 gas)
    ✓ should compute correct fee (169568 gas)
    ✓ should not be able to withdraw fees before the minimum is met (29700 gas)
    ✓ should be able to withdraw fees back to L1 once the minimum is met (183888 gas)

  Liquidity Pool Test
    ✓ should deposit 10000 TEST ERC20 token from L1 to L2 (295186 gas)
    ✓ should transfer L2 ERC20 TEST token from Bob to Alice and Kate (290759 gas)
    ✓ should add 1000 ERC20 TEST tokens to the L2 token pool (151222 gas)
    ✓ should register L1 the pool (53536 gas)
    ✓ should register L2 the pool (156384 gas)
    ✓ shouldn't update the pool (137208 gas)
    ✓ should add L1 liquidity (34360 gas)
    ✓ should add L2 liquidity (345658 gas)
    ✓ should fast exit L2 (282743 gas)
(node:20) DeprecationWarning: expectEvent.inLogs() is deprecated. Use expectEvent() instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
    ✓ should withdraw liquidity (235743 gas)
    ✓ shouldn't withdraw liquidity (130355 gas)
    ✓ should withdraw reward from L2 pool (103261 gas)
    ✓ should withdraw reward from L1 pool (66543 gas)
    ✓ shouldn't withdraw reward from L2 pool (110140 gas)
    ✓ should fast onramp (241152 gas)
    ✓ should revert unfulfillable swap-offs (543378 gas)
    ✓ should revert unfulfillable swap-ons (370960 gas)
```

```
      ✓ Should rebalance ERC20 (364320 gas)
      ✓ Should revert rebalancing LP (152867 gas)
      ✓ Should rebalance OMGLikeToken (305734 gas)
      ✓ should be able to pause L1LiquidityPool contract (152867 gas)
      ✓ should be able to pause L2LiquidityPool contract (279411 gas)
     - the DAO should be able to configure fee for L2LP
      ✓ should fail configuring L2LP fee for non DAO (26175 gas)
     - the DAO should be able to configure fee for L1LP
      ✓ should fail configuring L1LP fee for non DAO (26175 gas)
    OVM_ETH tests
        ✓ should add L1 liquidity (26175 gas)
        ✓ should add L2 liquidity (272681 gas)
        ✓ should fast exit L2 (217186 gas)
        ✓ Should rebalance ETH (276310 gas)
        ✓ Should revert rebalancing LP (152411 gas)
        ✓ should withdraw liquidity (231803 gas)
        ✓ should withdraw reward from L2 pool (133390 gas)
        ✓ should fast onramp (229294 gas)
        ✓ should revert unfulfillable swap-offs (444493 gas)
        ✓ should revert unfulfillable swap-ons (348483 gas)
    Relay gas burn tests
        ✓ should not allow updating extraGasRelay for non-owner (203221 gas)
        ✓ should allow updating extraGasRelay for owner (291407 gas)
        ✓ should be able to fast exit with correct added gas (568905 gas)

  Fast Messenge Relayer Test
      ✓ should send message from L1 to L2 (155201 gas)
      ✓ should QUICKLY send message from L2 to L1 using the fast relayer (227412 gas)

  Native ETH value integration tests
      ✓ should allow an L2 EOA to send to a new account and back again (295574 gas)
    calls between OVM contracts with native ETH value and relevant opcodes
        ✓ should allow ETH to be sent (31901 gas)
        ✓ should revert if a function is nonpayable
        ✓ should allow ETH to be sent and have the correct ovmCALLVALUE
        ✓ should have the correct ovmSELFBALANCE which includes the msg.value (167351 gas)
        ✓ should have the correct callvalue but not persist the transfer if the target reverts
        ✓ should look like the subcall reverts with no data if value exceeds balance
        ✓ should preserve msg.value through ovmDELEGATECALLs (99657 gas)
        ✓ should have correct address(this).balance through ovmDELEGATECALLs to another account (99657 gas)
        ✓ should have correct address(this).balance through ovmDELEGATECALLs to same account
        ✓ should allow delegate calls which preserve msg.value even with no balance going into the inner call (221162 gas)

  Native ETH Integration Tests
      ✓ receive (373513 gas)
      ✓ depositETH (305363 gas)
      ✓ depositETHTo (321024 gas)
      ✓ deposit passes with a large data argument (1913471 gas)
      ✓ depositETH fails with a TOO large data argument (1745459 gas)
      ✓ withdraw (1883110 gas)
      ✓ withdrawTo (275809 gas)
      ✓ deposit, transfer, withdraw (449941 gas)
    estimateGas
        ✓ Should estimate gas for ETH withdraw (137711 gas)

  NFT Bridge Test
      ✓ should deposit NFT to L2 (1995870 gas)

  OVM Context: Layer 2 EVM Context
      ✓ enqueue: L1 contextual values are correctly set in L2 (963275 gas)
      ✓ should set correct OVM Context for `eth_call` (105000 gas)
      ✓ should return same timestamp and blocknumbers between `eth_call` and `rollup_getInfo`

  Dao Action Test
    Config fee L2 LP
        ✓ should delegate voting rights (330480 gas)
0x01
        ✓ should create a new proposal to configure fee (1775557 gas)
         waiting for voting period to end...
        ✓ should cast vote to the proposal and wait for voting period to end (5578539 gas)
        ✓ should queue the proposal successfully (343750 gas)
        ✓ should execute the proposal successfully (312141 gas)
    Config fee L1 LP
        ✓ should create a new proposal to configure fee (1879834 gas)
         waiting for voting period to end...
        ✓ should cast vote to the proposal and wait for voting period to end (5625777 gas)
        ✓ should queue the proposal successfully (346548 gas)
        ✓ should execute the proposal successfully (391229 gas)

  Queue Ingestion
      ✓ should order transactions correctly (645754 gas)

  Basic RPC tests
    eth_sendRawTransaction
        ✓ should correctly process a valid transaction (157933 gas)
        ✓ should not accept a transaction with the wrong chain ID (21000 gas)
        ✓ should not accept a transaction without a chain ID (21000 gas)
        ✓ should accept a transaction with a value (42000 gas)
        ✓ should reject a transaction with higher value than user balance (21000 gas)
        ✓ should correctly report OOG for contract creations (21000 gas)
    eth_call
        ✓ should correctly identify call out-of-gas (21000 gas)
        ✓ should correctly return solidity revert data from a call (21000 gas)
        ✓ should produce error when called from ethers (21000 gas)
        ✓ should correctly return revert data from contract creation (21000 gas)
        ✓ should correctly identify contract creation out of gas (21000 gas)
        ✓ should allow eth_calls with nonzero value (273008 gas)
    eth_getTransactionReceipt
        ✓ correctly exposes revert data for contract calls (173924 gas)
        ✓ correctly exposes revert data for contract creations (83737 gas)
        ✓ includes L1 gas price and L1 gas used (83164 gas)
    eth_getTransactionByHash
        ✓ should be able to get all relevant l1/l2 transaction data (42000 gas)
    eth_getBlockByHash
        ✓ should return the block and all included transactions (42000 gas)
    eth_getBlockByNumber
        ✓ should return the same result when new transactions are not applied (21000 gas)
    eth_getBalance
        ✓ should get the OVM_ETH balance (21000 gas)
    eth_chainId
        ✓ should get the correct chainid (21000 gas)
    eth_estimateGas
        ✓ gas estimation is deterministic (21000 gas)
        ✓ should return a gas estimate for txs with empty data (21000 gas)
        ✓ should fail for a reverting call transaction (21000 gas)
        ✓ should fail for a reverting deploy transaction (21000 gas)
    rollup_gasPrices
      - should return the L1 and L2 gas prices

  stress tests
    L1 => L2 stress tests
        ✓ 10 L1 => L2 transactions (serial) (1194388 gas)
        ✓ 10 L1 => L2 transactions (parallel) (1194508 gas)
    L2 => L1 stress tests
        ✓ 10 L2 => L1 transactions (serial) (910080 gas)
        ✓ 10 L2 => L1 transactions (parallel) (910080 gas)
    L2 transaction stress tests
        ✓ 10 L2 transactions (serial) (418780 gas)
        ✓ 10 L2 transactions (parallel) (418780 gas)
    C-C-C-Combo breakers
        ✓ 10 L2 transactions, L1 => L2 transactions, L2 => L1 transactions (txs serial, suites parallel) (2474132 gas)
        ✓ 10 L2 transactions, L1 => L2 transactions, L2 => L1 transactions (all parallel) (2474168 gas)

  Whitelist
    when the whitelist is disabled
        ✓ should be able to deploy a contract (112612 gas)
    when the whitelist is enabled
        ✓ should fail if the user is not whitelisted (112612 gas)
        ✓ should succeed if the user is whitelisted (112612 gas)

·-----------------------------------------|---------------------------|--------------|-----------------------------·
|          Solc version: 0.8.9            ·  Optimizer enabled: true  ·  Runs: 10000  ·  Block limit: 6718946 gas  │
··········································|···························|··············|······························
|  Methods                                ·               100 gwei/gas               ·      4694.34 usd/eth       │
··················|·······················|·············|·············|··············|·············|················
|  Contract       ·  Method               ·  Min        ·  Max        ·  Avg         ·  # calls    ·  usd (avg)    │
··················|·······················|·············|·············|··············|·············|················
|  ERC20          ·  approve              ·      24938  ·      44174  ·       39874  ·          8  ·      18.72    │
··················|·······················|·············|·············|··············|·············|················
|  ERC20          ·  transfer             ·      53524  ·      53536  ·       53533  ·          8  ·      25.13    │
··················|·······················|·············|·············|··············|·············|················
|  SimpleStorage  ·  setValueNotXDomain   ·      36958  ·      86158  ·       41644  ·         42  ·      19.55    │
··················|·······················|·············|·············|··············|·············|················
|  ValueCalls     ·  simpleSend           ·          -  ·          -  ·       31901  ·          2  ·      14.98    │
··················|·······················|·············|·············|··············|·············|················
|  Deployments                            ·                                          ·  % of limit  ·              │
··········································|·············|·············|··············|·············|················
|  OVMContextStorage                      ·          -  ·          -  ·      226434  ·      3.4 %  ·     106.30    │
··········································|·············|·············|··············|·············|················
|  OVMMulticall                           ·          -  ·          -  ·      416898  ·      6.2 %  ·     195.71    │
··········································|·············|·············|··············|·············|················
|  Reverter                               ·          -  ·          -  ·      136933  ·        2 %  ·      64.28    │
··········································|·············|·············|··············|·············|················
|  SendETHAwayAndDelegateCall             ·          -  ·          -  ·      221162  ·      3.3 %  ·     103.82    │
··········································|·············|·············|··············|·············|················
|  SimpleStorage                          ·          -  ·          -  ·      237770  ·      3.5 %  ·     111.62    │
```

```
·······································|·············|·············|···············|··············|··············
|  ValueCalls                          ·          -  ·          -  ·       496209  ·       7.4 %  ·      232.94  |
·······································|·············|·············|···············|··············|··············
|  ValueContext                        ·          -  ·          -  ·        99657  ·       1.5 %  ·       46.78  |
·--------------------------------------|-------------|-------------|---------------|--------------|-------------·

  122 passing (15m)
  4 pending
```

Code Coverage

The repository that was in scope for this audit does have code coverage enabled for the smart contracts specific to Optimism. However, the contracts specific to Boba do not have code coverage enabled when running tests. Therefore, we cannot include any coverage data in this report. We recommend that code coverage is enabled for the Boba-specific contracts and be kept above 90% at all times.

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

```
4122f5e08bd25a4968a71ccbb45b01b2b8a8f48e3a26bf85b295ad69c9be516c ./packages/message-relayer/test/test-contracts/MockL2CrossDomainMessenger.sol

e9c6cc8d8a80db9c62d2f1c474e105a5755974e924b415ce8efd8ecffcafa668 ./packages/contracts/contracts/standards/AddressAliasHelper.sol

81dd25226789a2e075543f01478f132936454f81b080b7d1eda7285c588503ae ./packages/contracts/contracts/standards/L2StandardERC20.sol

d530eb5975c9ccebfb4cc754e98e4f04ac4444b4f03e12fd767f8710ec2b8a4b ./packages/contracts/contracts/standards/IL2StandardERC20.sol

c0fd4f5a1de60e24725510cbddd763f1c806d44dd404cc0b6509801287cbc30f ./packages/contracts/contracts/L1/messaging/IL1StandardBridge.sol

dc6a660a357c4f755a9e2b41d9181e69450826ee37e65bfe5c0405a7111031f4 ./packages/contracts/contracts/L1/messaging/L1MultiMessageRelayer.sol

ad1265c3f40fc347b1a568e80ddd1b2c5bdba9d15d37f5968c50a75d8d9175cd ./packages/contracts/contracts/L1/messaging/L1StandardBridge.sol

7dad4feed34ef8b868ac55fe52e339cfd0eaa061590540236ea62c95c22592c5 ./packages/contracts/contracts/L1/messaging/IL1CrossDomainMessenger.sol

a82eed000b8cdbfd9856923f6f25e1d69f5167545d4cf9b1e347a8959baf6d03 ./packages/contracts/contracts/L1/messaging/IL1ERC20Bridge.sol

d8f97fbcc57e856d5f195a994a29119ac01bde28a0f9301b4dffbce62ecc6b37 ./packages/contracts/contracts/L1/messaging/L1CrossDomainMessenger.sol

fe3860c8049b75ee705ba25b7038e4bf4fc1949db5db147911ce13b796b0573a ./packages/contracts/contracts/L1/rollup/StateCommitmentChain.sol

4c159074e7a14115a2bee61236637f2464f09305684dd3912b325d57f70d1f1d ./packages/contracts/contracts/L1/rollup/IChainStorageContainer.sol

ad4bac982deff3e4c1005d7016f49d178c5c4b30f6fe87ddca999863eb0cfb1e ./packages/contracts/contracts/L1/rollup/IStateCommitmentChain.sol

ed07a12750e7938106ebfc14d6c517cb361465322c9abeb06752faab77e2d142 ./packages/contracts/contracts/L1/rollup/CanonicalTransactionChain.sol

951758ddeeb7eeb71e22ae3075b727c3b5018946b8b5770b37892700ec36942c ./packages/contracts/contracts/L1/rollup/ChainStorageContainer.sol

3282a9edbab97cb4d19842095ad211de4b921d1132a1bf6262fa9d88ec8ddc28 ./packages/contracts/contracts/L1/rollup/ICanonicalTransactionChain.sol

7b63d08c4651ade0b6e6324d84163571eed88d9490a19aa37b1707429501ad19 ./packages/contracts/contracts/L1/verification/IBondManager.sol

ab9a9f566a07bb95464f30df70259a58e1989afeba3860830a6ccbbba75fb9a1 ./packages/contracts/contracts/L1/verification/BondManager.sol

79e225483c8ecf2c488ab96d138ee73e5abd352d7c310059a20f75fc8f55af8b ./packages/contracts/contracts/test-
libraries/standards/TestLib_AddressAliasHelper.sol

808d452fefea5faf4cbb990a3b11decfb190384e6299e0c4003a7f08c8fd5f61 ./packages/contracts/contracts/test-libraries/trie/TestLib_MerkleTrie.sol

bf20e17519a7be6ff3fff4cae2dabe05d37889e82b501ab89eb6ee2c7a353859 ./packages/contracts/contracts/test-
libraries/trie/TestLib_SecureMerkleTrie.sol

28956d2dd741dc4196651a88a212a3ae94a95279ed2203573f78c4e04666d58e ./packages/contracts/contracts/test-libraries/rlp/TestLib_RLPWriter.sol

5ffa03d7bcbdbd9e249423ec015ee841f778a61c476553261f4ad0b6f31afc0c ./packages/contracts/contracts/test-libraries/rlp/TestLib_RLPReader.sol

edfb4bd4668e403648ef1cb5a1df71d3e2124d2f75e240be027e8a056282a42b ./packages/contracts/contracts/test-libraries/utils/TestLib_MerkleTree.sol

ea56d12bc50cb0d490916cecae191de0435775e249f47d8fee1cda429ec200b6 ./packages/contracts/contracts/test-libraries/utils/TestLib_BytesUtils.sol

510b918ec84f04a126d36d08ec5f26684a909ce0aa0a585a9eef0b5e0881f87e ./packages/contracts/contracts/test-libraries/utils/TestLib_Buffer.sol

7d825d8f068890434da157004cca8d22552838231ba3d73d76092c60510a1145 ./packages/contracts/contracts/test-libraries/utils/TestLib_Bytes32Utils.sol

0806a93d7228bac1c81701cec20ca666b38fd59013dbde81bf4298a7bd0cf8d8 ./packages/contracts/contracts/test-libraries/codec/TestLib_OVMCodec.sol

5e30f9eec56d7144924e6bb0af8db2587f900cd995a3771b06abd2051b7f4c4f ./packages/contracts/contracts/test-helpers/Helper_SimpleProxy.sol

0d55e723d2670abcb96516dcc8c6e8c9dc78892c1b992b47d7198d9b36f032b6 ./packages/contracts/contracts/test-helpers/Helper_GasMeasurer.sol

b32736b43813349028afeee3a34deb4ebd5f731fa098f57cc6ab9f7dcf50b519 ./packages/contracts/contracts/test-helpers/TestERC20.sol

05301fa6f3d722b5a568ceb11ef6cb07112ccbee8d1dfd5cbd44b01b6ba8fc63 ./packages/contracts/contracts/chugsplash/L1ChugSplashProxy.sol

8fbceb93e5c2d37af11fa7f88f36c92921601869c3a95a9f0f5c380707d0ad9d
./packages/contracts/contracts/chugsplash/interfaces/iL1ChugSplashDeployer.sol

47bd49404be8d5a2498589f39148302baa5c652ae70d2205f6402600f84d13d6 ./packages/contracts/contracts/libraries/resolver/Lib_AddressManager.sol

63397cb1c3369a584be65649880bdccff2fc3df3b0d473dbf13eff7b1a837de3 ./packages/contracts/contracts/libraries/resolver/Lib_AddressResolver.sol

4968ef7974ee0546007e16f55c45c16a782d93601ea2632558145deae66db461
./packages/contracts/contracts/libraries/resolver/Lib_ResolvedDelegateProxy.sol

f82b762788c2f81837e66aab2006c0e546650bd6f1e74ce6b3fd7606e4adea76 ./packages/contracts/contracts/libraries/trie/Lib_SecureMerkleTrie.sol

430dab67285eaf172d624a7bc739a6cf63cd2f55d4c1d2acde4bee7337fcb541 ./packages/contracts/contracts/libraries/trie/Lib_MerkleTrie.sol

99c2f549e3eed3bfa955258e8fe3ddf89166620b8b0c5640ec0cc131e442f85e ./packages/contracts/contracts/libraries/rlp/Lib_RLPReader.sol

925953c2a09f8ebfbe1a0a6a0a0c6b8503314f69fb08b92a06673ade7de974ba ./packages/contracts/contracts/libraries/rlp/Lib_RLPWriter.sol

980bbe077572b8ba8384eda744debe801b52b4fc7b1606cb56655736d30c66b3 ./packages/contracts/contracts/libraries/utils/Lib_Bytes32Utils.sol
```

```
86103e721bef621752b40f38829865812aae205fa84fe9eb02cb890359b4f8b2 ./packages/contracts/contracts/libraries/utils/Lib_BytesUtils.sol
6c2c0051ee56b0da93718fc0406fbc07419d9e4d3c08403d393297440c909840 ./packages/contracts/contracts/libraries/utils/Lib_Buffer.sol
8b07c2cb5d4bc8e938b0587cfdfe7d7fe56a3f86b4f1d1e14e7f7dc6f432ab79 ./packages/contracts/contracts/libraries/utils/Lib_MerkleTree.sol
1446870b2ef35d6fbef8f4e8b52cec4503822659b43f27b3747b9e6b09054923 ./packages/contracts/contracts/libraries/codec/Lib_OVMCodec.sol
f1cc2103875b0ef9dc9818942c090765c7ed510f76ecb7a71d34dcad7c76576e ./packages/contracts/contracts/libraries/constants/Lib_DefaultValues.sol
63b3a1a3d396d7c707b11499891ba02ac0cf9aa2666536e12f1cbc21f8df0ddc ./packages/contracts/contracts/libraries/constants/Lib_PredeployAddresses.sol
d50d07398366719cee9e0c2d65d013ad141fa1213c72a33dcfec161d9dc4574d ./packages/contracts/contracts/libraries/bridge/ICrossDomainMessenger.sol
0e51e416dc508d6e4bd8448fd23e859475013bd0569a2f6f92deb8b34cfd45a3 ./packages/contracts/contracts/libraries/bridge/CrossDomainEnabled.sol
27e870780b12d9d2160870513902e6ab468f88e75ba277d1b213a7f0034107c8 ./packages/contracts/contracts/libraries/bridge/Lib_CrossDomainUtils.sol
5e4f9d61baab05f06e83026e92c363ba36850df62b18a2fbf052876c714052a8 ./packages/contracts/contracts/L2/messaging/L2StandardTokenFactory.sol
a124ba86a7d7b0fac26b1981213b7e88ea9eac0e5491e4bb54d18ae4ad156f57 ./packages/contracts/contracts/L2/messaging/L2CrossDomainMessenger.sol
a2096f4a0d4d0e7418ee0f8628d866e486bfc9d5fbfa98639deb5c0e4de91bc3 ./packages/contracts/contracts/L2/messaging/IL2ERC20Bridge.sol
a775463a4ad4f800325442fe6ee63872ddc318909a942ba23c3dcbd5df3b097d ./packages/contracts/contracts/L2/messaging/IL2CrossDomainMessenger.sol
10c6dc18573d2f08ca0e33dfcfb818c92e54d611687eb2fb51b8097d7b3f26e9 ./packages/contracts/contracts/L2/messaging/L2StandardBridge.sol
d645cdfd1d4ac0053e014dd93df34064444bdad3f65b7d4188173d2bd955823d ./packages/contracts/contracts/L2/predeploys/WETH9.sol
5f2b1695a90d4f9fa99c53a03b8b111fbf98ac473ae84d0b3c757858645a59dd ./packages/contracts/contracts/L2/predeploys/OVM_SequencerFeeVault.sol
04dc799cd221282f1f3ca48a8d1125ee35bfcf1feb4028f63a21e75750ad42ef ./packages/contracts/contracts/L2/predeploys/OVM_L2ToL1MessagePasser.sol
37513c5d16e0730d4a7688bc787473d72e7450b41b849b47b699ab6154e3e7a8 ./packages/contracts/contracts/L2/predeploys/iOVM_L1BlockNumber.sol
b28965b9a5e2e00c69522910af7a94c36a88198aba79b542cdf28d33e77e7731 ./packages/contracts/contracts/L2/predeploys/iOVM_L2ToL1MessagePasser.sol
e97611e08234b70e90b0daaacfde014a9bea8b44755263d3edd2abe88c7ed4f2 ./packages/contracts/contracts/L2/predeploys/OVM_ETH.sol
b3e18156e13b518f6eca38cc08813c83d3b6f87dee740badec13c8a6c67d7225 ./packages/contracts/contracts/L2/predeploys/OVM_GasPriceOracle.sol
1c62c433ae59a1d667ce5b368861e1c9c1a4454b17e2bea2fdffda0e3cc81338 ./packages/contracts/contracts/L2/predeploys/OVM_DeployerWhitelist.sol
295af1061e14fd15dfd7754a35b4b5f4da873fd2337eaeae0360b9a8a1be6e29 ./packages/boba/contracts/contracts/ERC721Registry.sol
4fb4077354559b7c4a5d34ad8f9c09da4549326cc3a03103fa8ae369e713552d ./packages/boba/contracts/contracts/ERC2470.sol
3c6d9984b463d0d83a0fd95efd60e1aaac59419b6d7dfcb2c3f10c900f3fd73d ./packages/boba/contracts/contracts/TokenPool.sol
4c572303ae9d838c6a653b65665de03e60df2657a529c94a5f13c03f00d9f052 ./packages/boba/contracts/contracts/L1MultiMessageRelayerFast.sol
2694724def47a9ceb4399a772ecef47c9b113a70200f160f19ff76041a5f557c ./packages/boba/contracts/contracts/L1CrossDomainMessengerFast.sol
eaa997a930929003b32ea8d8cf7b87e80c43c6676acb2b73e879f5da237f3df0 ./packages/boba/contracts/contracts/ERC721Genesis.sol
a2ee848e302070a1428563b3f9bea1e5f93437b56d6101f8bc0a176045cadeda ./packages/boba/contracts/contracts/AtomicSwap.sol
b6fe574d1712038122a0e7f67c47f127b27ba7148b9f3c603b30175e29815d87 ./packages/boba/contracts/contracts/standards/L2GovernanceERC20.sol
2f482bad31bc0bbc500ac730f0a2eb0c70da55a0cc19717486b628e73499a487 ./packages/boba/contracts/contracts/standards/IL2StandardERC721.sol
c87c75fec33109ef936ad629ee8b2ac7ec44d379fd413f2239e571e402455054 ./packages/boba/contracts/contracts/standards/L2StandardERC721.sol
80b35ca5fa8b2f4bb4a886aa9e4c17ea9952997214b6e7ee75241254828707e4 ./packages/boba/contracts/contracts/test-helpers/OMGLikeToken.sol
3b3a550a81c0fa8e5808bc24676c5e1b46475ff9653282c70ee485293ed76e57 ./packages/boba/contracts/contracts/test-helpers/L1ERC20.sol
06a225c00e6b783cb66eef8cb1b03bf9aba457e474fefa1647b95e88b4be4546 ./packages/boba/contracts/contracts/test-helpers/L1ERC721.sol
2d949d5b7ad84e84325f8ad16cac82392b2b75fdf48f452204dc986295b2709f ./packages/boba/contracts/contracts/test-helpers/Message/L2Message.sol
87cea07526430288cff806abce927eac2c5189d1bc9bebeb8585f62cd04cb7e3 ./packages/boba/contracts/contracts/test-helpers/Message/L1Message.sol
0d3dce416fbd8290f4b88b8460d2bb7cb7673c3033f416dd064b2136195ffc5d ./packages/boba/contracts/contracts/bridges/L1NFTBridge.sol
88fa3226a81482edabcaa4a8b781a27177a90c86549aae277d444d08a7aad66d ./packages/boba/contracts/contracts/bridges/L2NFTBridge.sol
20d024aca4a027fe8f5deab7d874ddff63c2bef5bbc59272ea8e10765c059c9e ./packages/boba/contracts/contracts/bridges/interfaces/iL1NFTBridge.sol
204f93554b3ffaea1488ab40a3374f02cb6357709b6b025f5ba281ab6a4482d7 ./packages/boba/contracts/contracts/bridges/interfaces/iL2NFTBridge.sol
31f68149ca0ab27ad96a815a1b35dbcfe4410e610c5ca86afc7d96963abe1137 ./packages/boba/contracts/contracts/libraries/CrossDomainEnabledFast.sol
e660ad44787d6a13028cea42fdfbf05cf6dbc94dff2df9c4ad68944fc733af57 ./packages/boba/contracts/contracts/libraries/Lib_ResolvedDelegateProxy.sol
c73fcff372d8199b723463fbb403c43a7ed99a80dc0ec64a245516492e02e94b ./packages/boba/contracts/contracts/DAO/governance/SafeMath.sol
32675f996dcf2367569c426a9e23f14109580361c43628e83eff35648ff2ebc1 ./packages/boba/contracts/contracts/DAO/governance/Timelock.sol
73f3e78d9dde5d5a41a779d789d3e6bbeb115328aa6484d49b4fab78703b15e1
./packages/boba/contracts/contracts/DAO/governance/GovernorBravoInterfaces.sol
c1adf18448eb8e071ed05d2b0a50037a76a83f2e60c235471c9fcd5371123766 ./packages/boba/contracts/contracts/DAO/governance/GovernorBravoDelegate.sol
489be8a9c67a544ed7538d1ffb5e53cd6440ef4c33ce40e1fa27d3e5f722b09c ./packages/boba/contracts/contracts/DAO/governance/GovernorBravoDelegator.sol
08c65a08274fe8228b42216536861875622265ce16cdf1c99026dd2b52cc8b373 ./packages/boba/contracts/contracts/DAO/governance-token/BOBA.sol
518aad3fc2767ee5896d2d9a14361e99bc3cf6823727c37b719594397c17988f ./packages/boba/contracts/contracts/LP/L2LiquidityPool.sol
554867f0b64e26cf59153926869b5f0775587e2eea813197cfea2087750a0e8f ./packages/boba/contracts/contracts/LP/L1LiquidityPool.sol
60230cf3aaaa8e3543d8bd1de922440cbaca3cc591c4a4d1f0bd9822212c1afc ./packages/boba/contracts/contracts/LP/interfaces/iL2LiquidityPool.sol
02cf8d6300a4b14796e247d7be5d051b198228ec3ae0c6e1f833c268b7b4afc8 ./packages/boba/contracts/contracts/LP/interfaces/iL1LiquidityPool.sol
```

Tests

```
e978d46089dfd91eec7a50fe068f491203fcf1da05597d547983a5783b85ae78 ./integration-tests/.eslintrc.js
937ef704e4ecab548e4c56ee3a21bf387bb3c240592bf9b21150d3db5ba097ac ./integration-tests/hardhat.config.ts
7fbb4f9fadee7ecc76c5300cfec295838e608e188e9d114dfab29f5ab43ea3c8 ./integration-tests/sync-tests/1-sync-verifier.spec.ts
669f50ddda95c1f0614e8678d991d6df9e2407e5dd1c68e03ffe5d1b89e332f4 ./integration-tests/sync-tests/2-sync-replica.spec.ts
a000a489cdfaed89176c213789dd1ecfb9e62c99c6513341ac3afd7d85549f9f ./integration-tests/contracts/TestOOG.sol
2b7cdeaa04dd84aaee351defd57b581406cbe373b6299e72fe7816da5eb94c75 ./integration-tests/contracts/ERC20.sol
```

```
1ca3ea3ac83ce6ce53f51b953d498b3f3113b9666e2a6307b7614db13c0b366b ./integration-tests/contracts/OVMMulticall.sol
61d557943ce34018551fdd0a836525ff70f224b9445a1caf2f56fd9c19bc8be5 ./integration-tests/contracts/SimpleStorage.sol
e07dce0fb57295020654c809654dadf778a1a395823582de379ab912cb2d63cc ./integration-tests/contracts/Proxy.sol
79eecd2f5307cb98c8d3b8776c14f5bad6469f046fe1c7eddc888e137d88ae2b ./integration-tests/contracts/OVMContextStorage.sol
6d6c3e5c3c1fedaa0208461431b05d59d849d66821a1c1237c0dd5def75542a9 ./integration-tests/contracts/Reverter.sol
0b899b5c342ff8ffaf407611b2ea3c8d264d66390f7d27a359a389f1d693f8b5 ./integration-tests/contracts/ValueCalls.sol
d69c071086eb4b56ebb7f2923d695d1c8a15e0c9f223a81037ba8122306249af ./integration-tests/contracts/ConstructorReverter.sol
97e44358b64b7c4c3ab1f5d73b4a7ccaf002e08d7f6d42ba2c9eb7cbf83d2f17 ./integration-tests/test/native-eth-ovm-calls.spec.ts
551ff0ce04a0a4d2d1b2fc01dfc09bcebc589da1cb8a0e5587bcf9f684995eb7 ./integration-tests/test/basic-l1-l2-communication.spec.ts
a6eaf2e75eb9737a9e65dd41b835e9165a1cf96c29bb3d697fc9e21164e04c06 ./integration-tests/test/stress-tests.spec.ts
07861a63f51ef69610d1a09282fe03f103e97d45f7aaefa432236848e041666c ./integration-tests/test/fee-payment.spec.ts
2d866043b55cea8879f1e3095e6131dbecb02ab3015dbe82f2f2c150031d83a4 ./integration-tests/test/pool_dao_actions.spec.ts
f9189051dad321b22db843cec5f38b76f9f12b3c5459392749bacc100eda410a ./integration-tests/test/native-eth.spec.ts
76c110758a6ed38627b2af49e23f4598d98dc143ac54a71e7037c1f73c2cf299 ./integration-tests/test/erc20.spec.ts
e1b825c97bd25e908799048a1d0d522cba3848f5834e4fcb2d2e7d06bf2e4790 ./integration-tests/test/setup-docker-compose-network.js
3e956b1225ea16b845151d7c41d195868eac66eb0659fcf7f1ee7c61230364ee ./integration-tests/test/nft_bridge.spec.ts
4e5da0eb06a579295e3f53fd440434486ae6ecd6390e476e77a41da7d7b06ca8 ./integration-tests/test/basic_nft.spec.ts
5188ae1effeb39e4360f18576aa6a41ecfa8ee5d56961d8574a78b4f6ca68683 ./integration-tests/test/whitelist.spec.ts
90f4442d2e37ec36770ee654b9c9337034f4870ac462d352429ba1429df04955 ./integration-tests/test/rpc.spec.ts
1616a318add8e7a1b8cc690942a798b82a92ff3052a138e528f2dd348a697981 ./integration-tests/test/ovmcontext.spec.ts
15abf5c76035633b19434d3c901798bf348111416f8dc3058693c9f1f2704c12 ./integration-tests/test/mrf_lp.spec.ts
7a4e850c4b5f05afe2fd2661225da5b5cdf0e2fd3037ae8c3d769a4df19817d7 ./integration-tests/test/mrf_message.spec.ts
938c75523dbad4ba33c957f7c83ee62addd2980c989dda05ff7b47bd2a0f3407 ./integration-tests/test/queue-ingestion.spec.ts
850eee0265c46d5c2e11d2265ca7704127b913900edf3d840736aebd4f46b2c9 ./integration-tests/test/shared/stress-test-helpers.ts
b434d8889c63f315827e07bcf43ed8cc3a0005b89efdce75c629c0871dccf0e8 ./integration-tests/test/shared/utils.ts
b4e1f9ac83bf657f9281e535257c8bd237bc198fe2479fa573c90d1a4bde2a53 ./integration-tests/test/shared/watcher-utils.ts
1a6017eb2c32673dc54b38560c908b6e16f130a11368dda43e8a3410a1189c29 ./integration-tests/test/shared/env.ts
d615b7d0acc9805f90fab29c1d4f0e4da78f50ef5a3033e5e79d49ba677b2ab5 ./integration-tests/test/shared/docker-compose.ts
```

Changelog

- 2021-12-21 - Initial report
- 2022-01-21 - Reaudit report

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.