SOLIDIFIED

Audit Report for Hop Protocol - April 12, 2021

Summary

Audit Report prepared by Solidified covering the Hop Protocol smart contracts.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on March 22, 2021, and the results are presented here.

Audited Files

The source code has been supplied in the form of a GitHub repository:

https://github.com/hop-exchange/contracts

Commit number: `2d862089aa24e3956e419dbdf0de9011b0f76cad`

UPDATE: Fixes provided with commit number:
`82b65d891dd6a08232cca30391f7c3d290dd3e9e`

UPDATE: Further fixes provided with commit number:
`2730fd978b54ec150266b479798c6f1a988c795f`

The scope of the audit was limited to the following files:

```
contracts
├── admin
│   └── Timelock.sol
├── bridges
│   ├── Accounting.sol
│   ├── Bridge.sol
│   ├── HopBridgeToken.sol
│   ├── L1_Bridge.sol
│   ├── L1_ERC20_Bridge.sol
│   ├── L1_ETH_Bridge.sol
│   ├── L2_Bridge.sol
│   ├── L2_OptimismBridge.sol
│   ├── L2_UniswapWrapper.sol
│   └── L2_XDaiBridge.sol
├── interfaces
│   ├── IMessengerWrapper.sol
│   ├── IWETH.sol
│   ├── arbitrum
│   │   └── messengers
│   │       ├── IArbSys.sol
│   │       ├── IBridge.sol
│   │       ├── IGlobalInbox.sol
│   │       ├── IInbox.sol
│   │       ├── IMessageProvider.sol
│   │       └── IOutbox.sol
│   ├── optimism
│   │   └── messengers
│   │       ├── iOVM_BaseCrossDomainMessenger.sol
│   │       ├── iOVM_L1CrossDomainMessenger.sol
│   │       └── iOVM_L2CrossDomainMessenger.sol
│   └── xDai
│       └── messengers
│           └── iArbitraryMessageBridge.sol
├── libraries
│   └── MerkleUtils.sol
└── wrappers
    ├── MessengerWrapper.sol
    ├── OptimismMessengerWrapper.sol
    └── XDaiMessengerWrapper.sol
```

Intended Behavior

The smart contracts implement a protocol that allows users to move funds between different L2 solutions and between L1 and L2, without users having to wait for the L2 exit period for each cross-layer transaction.

Code Complexity and Test Coverage

Smart contract audits are an important step to improve the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of a smart contract system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**.

Note, that high complexity or lower test coverage does equate to a higher risk. Certain bugs are more easily detected in unit testing than a security audit and vice versa. It is, therefore, more likely that undetected issues remain if the test coverage is low or non-existent.

Criteria	Status	Comment
Code complexity	Medium	-
Code readability and clarity	High	-
Level of Documentation	Medium-High	-
Test Coverage	Medium-High	-

Test coverage report:

```
------------------------------------|----------|----------|----------|----------|----------------|
File                                |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
------------------------------------|----------|----------|----------|----------|----------------|
 admin/                             |        0 |        0 |        0 |        0 |                |
  Timelock.sol                      |        0 |        0 |        0 |        0 |... 287,289,294 |
 bridges/                           |       91 |    71.54 |    90.53 |    91.64 |                |
  Accounting.sol                    |      100 |      100 |      100 |      100 |                |
  Bridge.sol                        |      100 |    77.27 |    94.12 |    96.77 |        116,117 |
  HopBridgeToken.sol                |      100 |      100 |      100 |      100 |                |
  L1_Bridge.sol                     |      100 |    97.73 |      100 |      100 |                |
  L1_ERC20_Bridge.sol               |      100 |      100 |      100 |      100 |                |
  L1_ETH_Bridge.sol                 |        0 |        0 |        0 |        0 |       16,17,21 |
  L2_Bridge.sol                     |    95.52 |       65 |    95.65 |    95.71 |    335,336,342 |
  L2_OptimismBridge.sol             |       60 |        0 |      100 |       80 |             52 |
  L2_UniswapWrapper.sol             |    67.65 |       50 |       80 |    73.53 |... 117,118,119 |
  L2_XDaiBridge.sol                 |        0 |        0 |        0 |        0 |... 46,54,55,59 |
 interfaces/                        |      100 |      100 |      100 |      100 |                |
  IMessengerWrapper.sol             |      100 |      100 |      100 |      100 |                |
  IWETH.sol                         |      100 |      100 |      100 |      100 |                |
 interfaces/arbitrum/messengers/    |      100 |      100 |      100 |      100 |                |
  IArbSys.sol                       |      100 |      100 |      100 |      100 |                |
  IBridge.sol                       |      100 |      100 |      100 |      100 |                |
  IGlobalInbox.sol                  |      100 |      100 |      100 |      100 |                |
  IInbox.sol                        |      100 |      100 |      100 |      100 |                |
  IMessageProvider.sol              |      100 |      100 |      100 |      100 |                |
  IOutbox.sol                       |      100 |      100 |      100 |      100 |                |
 interfaces/optimism/messengers/    |      100 |      100 |      100 |      100 |                |
  iOVM_BaseCrossDomainMessenger.sol |      100 |      100 |      100 |      100 |                |
  iOVM_L1CrossDomainMessenger.sol   |      100 |      100 |      100 |      100 |                |
  iOVM_L2CrossDomainMessenger.sol   |      100 |      100 |      100 |      100 |                |
 interfaces/xDai/messengers/        |      100 |      100 |      100 |      100 |                |
  iArbitraryMessageBridge.sol       |      100 |      100 |      100 |      100 |                |
 libraries/                         |    72.41 |       80 |    66.67 |    72.41 |                |
  MerkleUtils.sol                   |    72.41 |       80 |    66.67 |    72.41 |... 40,81,84,97 |
 wrappers/                          |        0 |        0 |        0 |        0 |                |
  MessengerWrapper.sol              |        0 |        0 |        0 |        0 |          14,15 |
  OptimismMessengerWrapper.sol      |        0 |        0 |        0 |        0 |... 29,37,45,47 |
  XDaiMessengerWrapper.sol          |        0 |        0 |        0 |        0 |... 44,53,54,58 |
------------------------------------|----------|----------|----------|----------|----------------|
All files                           |     70.3 |       50 |    70.97 |    71.08 |                |
------------------------------------|----------|----------|----------|----------|----------------|
```

Issues Found

Solidified found that the Hop protocol contracts contain 1 critical issue, no major issue, 1 minor issue, in addition to 6 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as they refer to best practices.

Issue #	Description	Severity	Status
1	L1_ETH_Bridge.sol: _transferFromBridge() can be griefed	Critical	Resolved
2	Bridge.sol: timeSlotSize not bounded	Minor	Acknowledged
3	L1_ETH_Bridge.sol: incorrect revert messages	Note	-
5	No reentrancy guards	Note	-
6	Note on the economic model for challenges	Note	-
7	L2_Bridge.sol: l2CanonicalToken not used	Note	-
8	Duplicate SafeMath implementations	Note	-

Critical Issues

1. `L1_ETH_Bridge.sol:` `_transferFromBridge()` can be griefed

The function `_transferFromBridge()` sends ETH out and reverts on failure. This may cause permanent failure of certain kinds of `resolveChallenge()` flows. If the challenger is malicious, for example, the challenger address being a smart contract that reverts on the fallback function unless a variable is set, this flow can be used to ransom bonders and hold their credit hostage.

Recommendation

Consider using a withdrawal pattern.

Major Issues

No major issues have been identified.

Minor Issues

2. `Bridge.sol:` `timeSlotSize` not bounded

The variable `timeSlotSize` can be set to a value that causes the contract to just be unable to execute due to hitting the block gas limit.

Recommendation

Consider using a guard to make sure this cannot happen by accident.

Team Response

We have chosen to make no change. The reason being that this is a governance-controlled parameter and would only be changed with gas costs in mind. If governance is making this change maliciously, that is within our threat model.

Informative Notes

3. L1_ETH_Bridge.sol: incorrect revert messages

The revert message texts in functions `_transferFromBridge()` and `_transferToBridge()` relate to the ERC20 bridge contract.

Recommendation
Adjust the strings to reflect the correct contract.

4. No reentrancy guards

Given that bridge code calls out to messenger interfaces and calls out to token transfer functions which pass the full gas flow, it might be prudent to include reentrancy guards in critical functions just as a precaution, especially if in the future functionality is implemented that allows users to specify custom smart contract execution targets on the cross-layer tx.

Recommendation
Consider using reentrancy protection for critical functions.

5. Note on the economic model for challenges

Depending on the economics of the bonder and how the HOP exchange is used, a 10% deposit to lock up credit for the exit time duration might be too low, or worth it to the challenger to profit in some other way. There's the possibility of blackmailing a bonder if they really need the credit, and asking for ransom otherwise, they will submit a challenge, as well, and even losing 10% might not be enough to dissuade that depending on second-order effects that the bonder might experience by having that credit locked up for long periods of time, especially in the "future" cases where the bonder set is larger or open.

Recommendation
There is no specific recommendation for this beyond suggesting that the economic model could use a curve to calculate the challenger deposit percentage.

6. L2_Bridge.sol: l2CanonicalToken not used

The variable l2CanonicalToken is unused. This may be leftover from a previous version.

Recommendation
Consider removing unused variables.

7. Mixed compiler versions

The codebase uses a number of compiler versions, which may have different behavior.

Recommendation
We recommend locking the whole codebase to a single compiler version.

8. Duplicate SafeMath implementations

The codebase uses OpenZeppelin’s version of SafeMath in most places. However, `Timelock.sol` implements its own version.

Recommendation
Consider using a single version of the library.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Authereum / Hop Protocol or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.