SOLIDIFIED

Audit Report for Aztec - June 14, 2022

Summary

Audit Report prepared by Solidified covering the Aztec protocol Ethereum Bridge contract for **Liquity Bridge**.

The following report covers the **Liquity Bridge**.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 18 May 2022.

Audited Files

The source code has been supplied in the form of one public Github repository.

https://github.com/aztecProtocol/aztec-connect-bridges/

Commit Hash: `5a3766115b55dd3471ca981a26c8b8381f12fae7`

```
src
|-- bridges
|   -- liquity
      |-- interfaces
      |-- StakingBridge.sol
      |-- TroveBridge.sol
      |-- StabilityPoolBridge.sol
```

Intended Behavior

Smart contract responsible for depositing, managing and redeeming Defi interactions with the Aaave protocol by issuing an internal accounting token.

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 in 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	-
Test Coverage	Medium	-

Issues Found

Issue #	Description	Severity	Status
1.	Liquity - Partial-redemption breaks bridge accounting logic	Critical	Resolved
2.	Redistribute undercollateralized Troves breaks bridge accounting logic	Critical	Resolved
3.	Recover Mode Liquidation results in lost ETH if ratio is between 110%-150%	Critical	Resolved
4.	Uniswap token swap uses minimum output amount of 0	Note	-
5.	Uniswap token swap is always executed	Note	

Critical Issues

1. Liquity partial-redemption breaks bridge accounting logic

`Liquity` has the functionality of Redemption. Anyone can exchange 1 `LUSD` for 1$ `ETH`. This is a public function and callable by anyone.

(See `https://docs.liquity.org/faq/lusd-redemptions`)

A redemption uses the deposited `ETH` in the `Troves` starting with the riskiest troves (lowest collateralized).

The bridge has no way to improve the collateral-ratio of the Trove if the price of `ETH` starts to decrease. Each new deposit/withdrawal from the bridge only keeps the collateral-ratio constant.

A very low collateralization ratio is a likely scenario and only depends on the `ETH` price.

The bridge implementation covers the case where all the debt has been wiped out with redemptions.

However, it doesn't cover the scenario where the trove debt is only partially-redeemed.

The bridge logic issues a own accounting token which is built on the assumption that the debt never changes. The accounting token should match the amount exactly of debt.

In the partially-redeemed scenario the implemented formulas would start to behave incorrectly and users would not receive the correct amounts of `ETH`.

The repayment formula (exchange `LUSD` for `ETH`) only uses accounting token supply and deposited ETH collateral.

```
 outputValueA = (coll * inputValue) / this.totalSupply();
```

After a partial-redemption both the Troves collateral and debt are reduced. The bridge would first give users not enough ETH in exchange and afterward try to reduce the debt which is already zero.

Recommendation

The bridge accounting token logic can’t be built on the assumption that the Trove's debt will never change.

If a partial-redemption has happened the Trove will include less liquidity.

For returning all the deposited ETH the bridge needs to use the `redeemCollateral` functionality itself.

Resolved

The Aztec team implemented a new version of the bridge which is not built on the assumption that the internal debt and collateral value will never change.

New users get more tokens than `LUSD` after a partial redemption from the `_borrow` method. However, in the repayment case they need to provide the same amount of `LUSD` and token. (Because the `convert` method has only one `_inputValue` for both inputToken A and B).

The required `LUSD` amount can be higher than the actual borrowed amount. The bridge will return the additional `LUSD` but technically need to provide it in the first place.

Users need to be aware that if a partial redemption happens that they might need to acquire additional `LUSD` from the market to redeem their full collateral.

This can involve additional risk in case of low liquidity of `LUSD` in markets or unexpected prices.

2. Liquity Redistribute undercollateralized Troves breaks bridge accounting logic

The bridge implementation doesn't cover the 4.2 Redistribute undercollateralized Troves to other borrowers case.

(See here in the Liquity Whitepaper: `https://docsend.com/view/bwiczmy`)

If a redistribution happens and the Aztec bridge trove is affected the accounting/debt token will be not aware of the change.

In the worst case, if the debt increase is higher than the `coll` increase in the bridge troves: The repayment formula will start to behave incorrectly:

```
outputValueA = (coll * inputValue) / this.totalSupply();
```

The first users would get too much ETH which would also affect the ratio. (not constant)

In the worst case the ratio could get below/close to 110%.

Recommendation

The bridge accounting token logic can’t be built on the assumption that the Trove's debt will never change.

Resolved

The Aztec team implemented a new version of the bridge which is not built on the assumption that the internal debt and collateral value will never change.

3. Recover Mode Liquidation results in lost ETH if ratio is between 110%-150%

Liquity has a recovery mode if the total collateral ratio of all Troves falls below 150%.
Here all Troves below 150% can be liquidated. Further explanation in the last section in the Liquity whitepaper.

However, a liquidated Trove can get back some of the collateral ETH after the liquidation.

If the Trove ratio is in the 110%-150% range.

Let's say the collateral ratio is 140%. Liquity would only take 110% and the Trove o would still get the 30%.

This is handled in the Trove Manager (Line:414)

```
if (singleLiquidation.collSurplus > 0) {
   collSurplusPool.accountSurplus(_borrower, singleLiquidation.collSurplus);
}
```

The Trove receives a surplus which can be claimed with the `BORROWER_OPERATIONS.claimCollateral()` operation. It requires the trove owner to call it.

After such a liquidation the new Trove status will be `Status.closedByLiquidation`

The Trove bridge implementation can only call the `claimCollateral` if the status is `Status.closedByRedemption`

The current bridge implementation would be unable to claim the collateral in the worst case ~26% (40/150) of the ETH collateral would be locked/lost in the Liquity contracts.

Recommendation
The bridge should be able to call claimCollateral in the `Status.closedByLiquidation` as well. Afterwards the remaining ETH should be able to be withdrawn by the bridge users.

Resolved
The `claimCollateral` function can now be called in the `closedByLiquidation` state as well.

Major Issues

No issues found

Minor Issues

No issues found

Informational Notes

4. Uniswap token swap uses minimum output amount of 0

The Stability and Staking bridge are using Uniswap to exchange tokens. Currently, the minimum output amount is zero. A minimum output amount of zero can be used for sandwich attacks.

It might be worth considering a design where a minimum price can be calculated off-chain and passed as a parameter.

5. Uniswap token swap is always executed

It might make sense to add a threshold before using Uniswap. Currently, the swap might be performed for tiny amounts with higher gas costs than the actual received value.

The amount used for Uniswap depends on the frequency of the bridge usage together with the deposited amounts.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of Aztec 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 from legal and financial liability.

Oak Security GmbH