SOLIDIFIED

Audit Report for Aztec - September 2022

Summary

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

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

Process and Delivery

Independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 21 September 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: `f9f2554d63519c1b3288325f5edc50acaac2bbc5`

```
src
|-- bridges
|   -- dca
       |-- BiDCABridge.sol
       |-- SafeCastLib.sol
       |-- UniswapDCABridge.sol
```

Intended Behavior

The bridge implements a DCA (Dollar Cost Average) mechanism. Instead of buying a specific amount of tokens once, the buy happens in multiple equal chunks over a period of time regardless of the current price. In the implementation tokens are first deposited into the contract. After the next time period has passed orders can be matched first internally and if needed additional liquidity is provided or is requested from Uniswap.

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	High	-
Test Coverage	High	-

SOLIDIFIED

Audit Report for Aztec - September 2022

Issues Found

Issue #	Description	Severity	Status
1.	Out of Gas rebalanceAndFill	Minor	Resolved
2.	No Escape Mechanisms	Minor	Resolved
3.	Information Notes	Notes	-

Critical Issues

No issues found

Major Issues

No issues found

Minor Issues

1. Out of Gas _rebalanceAndFill

The loop in `_rebalanceAndFill` depends on the passed ticks. Theoretically, this could get so big that it won't fit in an Ethereum block anymore.

Recommendation:
A potential solution would be to offer two functions for rebalanceAndFillUniswap and rebalanceAndFill with an optional parameter endTick (which needs to be `endTick < nextTick && endTick > earliestTick`)

Resolved:
In the following commit `b0b4a19437569bb7b3e1b5fab5fb3cbb86f88bde`

2. No Escape Mechanisms

The `DCA` bridge does not provide any "escape" mechanisms in case `ORACLE` stops reporting prices. The funds will be locked forever if Uniswap doesn't have enough liquidity.

In case liquidity in Uniswap disappears (e.g. new version), then it is still possible to use the `rebalanceAndFill()` to finish the `DCA`, but in case there is no `ORACLE` , everything would stop.

Recommendation:
Consider a `withdraw` functionality in the `finalise` method for available funds. If the `ORACLE` is inactive for a critical period of time.

One additional safety mechanism for the oracle availability could be to limit the amount of ticks a user can pass to prevent ticks in the unforeseeable future.

Resolved:
In the following commit `2df45ffd5cd09dc37fd183bf093ad32771660243` by removing the `MAX_AGE` limit for the Oracle price.

This allows to fill all orders in contract with the public `rebalanceAndFill` method.

3. Informational Notes

getAccumulated simplification to reduce code complexity
For the bridge itself a method which only returns a flag if the entire interactionNonce position is accumulated or not would be sufficient.

Currently, the method returns the accumulated amount as well, which is not used on-chain.

The method could just return in case a not ready is detected, which would also simplify
`accumulated += sold == 0 ? 0 : (bought * tickAmount) / (sold + available);`

Helper function for nextTick
Consider a helper function for nextTick calculation, which appears multiple times
`uint256 nextTick = ((block.timestamp + TICK_SIZE - 1) / TICK_SIZE);`

Sanity Checks for protocolBought and protocolSold
Add sanity checks for the protocolBought and protocolSold calculations some cases should never happen. (Additional protection)

Constant for Fee basis points
Consider a constant for FEE basis points 10_000

Add a helper math function to divide with rounding up

A division with rounding up is required in multiple places in the codebase.

Audit Report for Aztec - September 2022

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