SOLIDIFIED

Audit Report for Aztec - May 11, 2022

Summary

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

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

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code. The debrief on 25 April 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: `ac2e7194b5887ea11a607b4cf8de0547b3d7fdd0`

```
src
|-- bridges
|   -- element
       |-- ElementBridge.sol
       |-- MinHeap.sol
       |-- interfaces
```

Intended Behavior

Smart contract responsible for depositing, managing and redeeming Defi interactions with the Element protocol

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	Medium	-
Level of Documentation	High	-
Test Coverage	High	-

Issues Found

Issue #	Description	Severity	Status
1	The amount of tokens received from Balancer exchange is not checked	Minor	Fixed
2	To improve code readability consider moving Nonce array and and MinHeap functionality to a single library	Note	-
3	Consider if Linked-List could be used to replace the MinHeap to reduce the code complexity	Note	-
4	Unused return value	Note	-
5	Misleading function name	Note	-

Critical Issues

No issues found

Major Issues

No issues found

Minor Issues

1. The amount of tokens received from Balancer exchange is not checked

The function `convert()` does not check the amount of tokens received from the Balancer exchange pool when doing a swap `IVault(balancerAddress).swap()`.

Recommendation
Consider checking the amount of tokens received from the Balance exchange.

Update May 5th, 2022:
Addressed with commit `9aee254f8232c0d4010352871b7819dca3c4dbca`.

Informational Notes

2. To improve code readability consider moving Nonce array and and MinHeap functionality to a single library

A tranche can be redeemed after a specific block.timestamp in the future. Currently, the bridge uses the leftover-gas to redeem tranches. The tranches should be redeemed in order of their expiration date.

This is achieved by maintaining a Min-Heap with an interactionNonce array for each Heap element.

Only the MinHeap has been moved into a separate library. It might make sense to move the interactionNonce array modification into the same library and offer high-level functions to get the next nonce or store a nonce.

3. Consider if Linked-List could be used to replace the MinHeap to reduce the code complexity

Instead of implementing a Min-Heap with a nonce queue for each Heap element, it could be a linked-list implementation for the nonces together with a pointer to the latest redeemed element.

We see this as a trade-off between code complexity and gas-optimisation.

An insert operation might require iterating over all nonces in the future (worst case). However, the redeem operation could be done in O (1) and it is not required to remove the element from the linked-list.

It depends on how many nonces/tranches are expected to be maintained by the bridge.

If the number is low, a simpler linked-list implementation could reduce the code-complexity and slight gas-cost increase might be acceptable.

4. Unused return value

Return values of `popInteraction` are unused.

5. Misleading function name

Function `checkNextExpiry` is not only checking the next expiry it is also modifying the contract data such as setting transactions to failed etc.

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