Bifrost

SLPx for Astar zkEVM

Smart Contracts Audit

Bifrost

Mar 8, 2024

Common Prefix

Overview

Introduction

Common Prefix was commissioned to perform a security audit on Bifrost's' SLPx for AstarZk smart contracts, at commit hash [e64ea84b955b57f8fd8602b38ba10bf46b307456]. The files inspected are the following:

```
├── AstarReceiver.sol
├── AstarZkSlpx.sol
└── DerivativeContract.sol
```

Protocol Description

Bifrost's SLPx pallet enables users to mint vTokens on a target chain within the Polkadot ecosystem without the need to manually bridge their initial tokens to Bifrost. Through XCM, the tokens are transferred to Bifrost, where vTokens are minted and subsequently transferred back to the target chain.

The focus of this audit pertains to the SLPx contracts specifically tailored for scenarios where the target chain is Astar zkEVM, a zkEVM chain that operates independently from Polkadot. As the XCM pallet cannot be utilized directly in Astar zkEVM, ASTR (the native token of the Astar parachain) serves as the intermediary asset and is transferred to Astar zkEVM as an OFT (On-Chain Fungible Token). These tokens within Astar zkEVM are then employed to mint vTokens. This process involves sending the tokens to the Astar parachain via Layer Zero. Subsequently, within the Astar parachain, the XCM pallet facilitates the transfer of ASTR to Bifrost, where vASTR tokens are minted. Users can then claim these vTokens and bring them back to AstarZk (*Note*: the team has changed this part and the user does not have to manually claim his tokens back to Astar, see Alleviation of issue LOW-1).

Similarly, users can redeem their vTokens held on AstarZk for the original tokens.

Disclaimer

Note that this audit does not give any warranties on the bug-free status of the given smart contracts, i.e. the evaluation result does not guarantee the nonexistence of any further findings of security issues. This audit report is intended to be used for discussion purposes only. Functional correctness should not rely on human inspection but be verified through thorough testing. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the project.

Findings Severity Breakdown

The findings are classified under the following severity categories according to the impact and the likelihood of an attack.

Level	Description
Critical	Logical errors or implementation bugs that are easily exploited and maylead to any kind of loss of funds
High	Logical errors or implementation bugs that are likely to be exploited and may have disadvantageous economic impact or contract failure
Medium	Issues that may break the intended contract logic or lead to DoS attacks
Low	Issues harder to exploit (exploitable with low probability), issues that lead to poor contract performance, clumsy logic or seriously error-prone implementation
Informational	Advisory comments and recommendations that could help make the codebase clearer, more readable and easier to maintain

Findings

Critical

No critical issues found.

High

No high issues found.

Medium

MEDIUM-1	Potential reversion of transactions due to high `_minAmount` in `OFTWithFee::sendAndCall`
Contract(s)	`AstarZkSlpx.sol`
Status	**Resolved**

Description

In the `AstarZkSlpx::mint` function, the `sendAndCall` function of the `OFTWithFee` contract is called with `_minAmount` set equal to the entire `_amount`. However, the `sendAndCall` function in `OFTWithFee` subtracts a dust amount from `_amount` to ensure it aligns with specific requirements. Consequently, the final `_amount` may be less than

`_minAmount`, causing the transaction to revert due to the condition in `sendAndCall` that `_amount` after deducting dust must exceed `_minAmount`.

Recommendation

We suggest adjusting the `_minAmount` parameter passed to `sendAndCall` to be a percentage of `_amount` or, ideally, set it to `_amount - dust`, where dust is computed as per the logic within the `OFTWithFee` contract. Alternatively, an extra argument `_minAmount` could be added in `mint`, allowing the user to decide the `_minAmount`.

Alleviation

The team fixed the issue at commit hash [3bf061b737b641363332b4410caf3034a90dfd21], setting a `_minAmount` equal to `_amount/2`.

Low

LOW-1	Lack of access control for `claimVAstr` and `claimAstr` functions
Contract(s)	`AstarReceiver.sol`
Status	**Resolved**

Description

The functions `claimVAstr` and `claimAstr` are currently callable by anyone. These functions facilitate the transfer of vASTR and ASTR, respectively, from the Astar parachain back to the originating address at AstarZk, i.e. to the address initially requested minting or redemption. However, there is currently no access control mechanism in place for these functions.

Consequently, anyone can invoke them and determine when these tokens are sent back to their respective owners on AstarZk.

While there is no risk of fund loss, and moreover whoever calls these functions will have to pay the fees for the cross-chain transfers, we believe that owners should have the privilege to decide when they wish to transfer their tokens back.

Recommendation

We recommend implementing access control for these functions, allowing them to be callable only when the argument `addr` matches `msg.sender`.

Alleviation

The team fixed the issue at commit hashes [823f33bb4584f2ad10d68a09ce047b4be23da14d]

and [b4d20d1a3ea3425b0da8683b3d39db0e93701bbe]. The claim functions are now callable only by a specific address (`scriptTrigger`) controlled by the team. The user does not have to claim back his tokens to Astar zkEVM and this task is taken by this address. This simplifies the UX, but of course requires a certain trust to the team, that it will actually call the claim functions and will not leave the user's tokens locked in the `DerivativeContract`.

Informational/Suggestions

INFO-1	Redundant function `getXtokensDestination`
Contract(s)	`AstarReceiver.sol`
Status	**Resolved**

Description

The `getXtokensDestination` function within the `AstarReceiver` contract is marked as internal but is never called by any other function within the contract. Consequently, its presence serves no purpose and adds unnecessary complexity to the contract.

Recommendation

We recommend removing the `getXtokensDestination` function from the contract to improve clarity.

Alleviation

The team fixed the issue at commit hash [a2740181b7bf0ca02c377f5cd4b3748a70bc90b5](), removing the redundant function.

INFO-2	Variables could be immutable
Contract(s)	`AstarReceiver.sol`, `AstarZkSlpx.sol`, `DerivativeContract.sol`
Status	**Resolved**

Description

The variables `astarZkSlpx`, `VASTR` and `destChainId` of the `AstarReceiver.sol`, `astrOFTWithFee`, `vAstrOFT` and `destChainId` of `AstarZkSlpx.sol` and `astarReceiver` of `DerivativeContract.sol` are set only in the constructor, therefore they could be immutable to deduce the contracts' storage and improve efficiency.

Alleviation

The team fixed the issue at commit hash [eac7bdb89996af89e976dbe8ac4c5d7f20c32099](), replacing the variables with constants.

INFO-3	Implement sanity checks in `AstarReceiver::onReceive` function
Contract(s)	`AstarReceiver.sol`
Status	**Resolved**

Description

Currently, the `_srcChainId` and `_srcAddress` arguments of the `AstarReceiver::onReceive` function are not utilized. However, incorporating sanity checks utilizing these parameters could enhance security and ensure that transactions originate from the expected source.

Recommendation

We suggest implementing sanity checks within the `onReceive` function to validate that the source chain corresponds to AstarZk and that the source address matches the OFT address associated with vASTR or ASTR on AstarZk.

Alleviation

The team fixed the issue at commit hash [9920deccd9306668043ce040de47bf6dbb98ccfb](), adding the suggested sanity checks.

About Common Prefix

Common Prefix is a blockchain research, development, and consulting company consisting of a small number of scientists and engineers specializing in many aspects of blockchain science. We work with industry partners who are looking to advance the state-of-the-art in our field to help them analyze and design simple but rigorous protocols from first principles, with provable security in mind.

Our consulting and audits pertain to theoretical cryptographic protocol analyses as well as the pragmatic auditing of implementations in both core consensus technologies and application layer smart contracts.