arbitrary./execution

MilkomedaM1

SIDECHAIN BRIDGE SECURITY ASSESSMENT

August 19, 2022

Prepared For:
Rob Kornacki

Prepared By:
John Bird, Jasper Clark

Changelog:

July 27, 2022	*Initial report delivered*
August 19, 2022	*Final report delivered*

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This report contains the results of Arbitrary Execution's assessment of the Milkomeda sidechain bridge smart contracts. The Milkomeda protocol enables Ethereum Virtual Machine (EVM) capabilities on non-EVM blockchains, allowing developers on other networks to leverage existing tooling in the EVM ecosystem. Milkomeda currently maintains an EVM-based sidechain for the Cardano blockchain.

Two Arbitrary Execution (AE) engineers conducted this review over a 1-week period, from July 15, 2022 to July 21, 2022. The audited commit was `aeb4efbf062980addbcd39571c295c1964e85032` in the main branch of the `dcSpark/milkomeda-validator` repository. The complete list of files in scope is located in Appendix B. These repositories were private at the time of the engagement, so hyperlinks may not work for readers without access. This engagement was a follow-up to a review of the protocol that took place in March 2022. This engagement focused on the files that have changed since the last engagement.

The team performed a detailed, manual review of the codebase with a focus on Milkomeda's `Multisig.sol` contract, which contains the functionality used by network validators. In addition to manual review, the team used Slither for automated static analysis.

The assessment resulted in one low-severity finding. The finding allows an attacker to bypass the sidechain bridge's proxy and call `initialize` on the implementation contract. No issues were identified with the code changes since the previous audit.

FIX REVIEW UPDATE

FIX REVIEW PROCESS

After receiving fixes for the findings shared with Milkomeda, the AE team performed a review of each fix. Each change was scrutinized to ensure that the core issue was addressed, and that no regressions were introduced with the fix. A summary of each fix review can be found in the *Update* section for a finding. For findings that the Milkomeda team chose not to address, the team's rationale is included in the update.

The Milkomeda team has fixed all major issues identified in the engagement. The full breakdown of fixes can be found in the Fixes Summary section.

AUDIT OBJECTIVES

AE had the following high-level goals for the engagement:

- Ensure Milkomeda's new changes have not introduced any security issues
- Identify smart contract vulnerabilities
- Evaluate adherence to development best practices

AE also set specific goals around Milkomeda's protocol:

- Ensure that the `validatorVotePeriod` variable is handled correctly in `Multisig.sol`
- Verify new structure members in `Types.sol` do not impact bridge behavior

System components including the `Multisig` and `Types` contracts will be covered in additional detail in the System Overview section.

OBSERVATIONS

Changes to the protocol since the last engagement are relatively minor. Additional logic was added to the `Multisig` contract to include a time delay for adding new validators to the sidechain network. Other changes included the addition of new struct members in `Types.sol`, and new events types and event emissions in several contracts.

Additionally, Foundry unit tests have been added to the existing suite of Truffle tests.

SYSTEM OVERVIEW

USER CATEGORIES

Users in the Milkomeda protocol are developers and end users who desire EVM-interoperability on a non-EVM chain.

SYSTEM COMPONENTS

Milkomeda allows users to send assets from a main chain to a bridge and receive funds on an EVM-compatible sidechain. The current deployment of Milkomeda allows users to wrap and unwrap assets from the Cardano blockchain.

MAIN CHAIN

The network which Milkomeda provides EVM interoperability to is referred to generically as the main chain. Users send assets from the main chain into Milkomeda's bridge and receive wrapped assets on Milkomeda's sidechain. Milkomeda currently supports Cardano as a main chain.

SIDECHAIN

The sidechain is Milkomeda's EVM-compatible network. Once users bridge assets from the main chain they can interact with smart contracts deployed on this network. Sidechain users receive wrapped assets in return for locking up assets on the main chain. The sidechain is a permissioned Hyperledger Besu network. Its behavior differs from Ethereum, primarily in its deployment process and consensus mechanisms.

Hyperledger Besu implements the IBFT 2.0 proof-of-authority (PoA) consensus protocol. Approved accounts, called validators, are responsible for validating blocks and transactions on the network.

ASSET TYPES

MAIN

A `MAIN` token is the native asset on the main chain side of the bridge (e.g. ADA).

WMAIN

`WMAIN` is shorthand for "wrapped main asset". This is a token on the sidechain that functions similarly to Ether. A `WMAIN` token can be used to pay fees on the sidechain. `WMAIN` is pre-minted to the bridge in the sidechain's network genesis file, and must be released to users by the network validators. In the present sidechain, the `WMAIN` token is wrapped ADA, and is referred to as `milkADA`.

SIDECHAIN BRIDGE

The sidechain bridge was the primary focus of this engagement. It is written in Solidity and is responsible for handling funds on the EVM side of Milkomeda. The bridge source code is broken up into multiple files for organization purposes:

- `Types.sol`: Holds type definitions for the other contracts
- `State.sol`: Holds all state variables for the bridge contracts
- `Multisig.sol`: Holds functions for adding and removing network validators, as well as functions for validators to vote on and execute transactions
- `TokenRegistry.sol`: Holds functions for managing the list of approved sidechain assets
- `Rewards.sol`: Holds logic for validator rewards
- `Proxy.sol`: The sidechain bridge's ERC-1967 compliant proxy based on OpenZeppelin's `Proxy.sol`
- `SidechainBridge.sol`: The implementation contract for the sidechain bridge

Extended descriptions for each file can be found in the protocol's documentation.

In short, two deployed contracts make up the sidechain bridge–the proxy contract, and the bridge implementation contract. The sidechain network validators interact with the bridge through the proxy.

VULNERABILITY STATISTICS

Severity	Count
Critical	0
High	0
Medium	0
Low	1
Note	0

FIXES SUMMARY

Finding	Severity	Status
L01	Low	Fixed in pull request #1260

FINDINGS

LOW SEVERITY

[L01] SIDECHAINBRIDGE IMPLEMENTATION DOES NOT PREVENT DIRECT INITIALIZATION

The SideChainBridge bridge implementation contract does not prevent the `initialize` function from being called directly.

This enables a malicious user to bypass the Proxy storage contract and call `initialize` on the implementation contract with their own initialization parameters. This could allow a malicious actor to use the implementation contract to masquerade as the Proxy contract. The impact of this is low, considering the initialization would act upon the implementation contract's storage, not the proxy's storage.

RECOMMENDATION

OpenZeppelin's `Initializable` library contains utilities for writing and protecting upgradeable contracts. Consider using the library and adding the `initializer` modifier to the `SidechainBridge` contract's `initialize` function. In conjunction with this modifier, the `_disableInitializers` function can be invoked in the `SidechainBridge` constructor to prevent direct initialization.

UPDATE

Fixed in pull request #1260 (commit hash `c75dd4cdfff5c5aa139944ae406d0137366028e8`). The implementation contract now sets `initialized = true` in its constructor.

APPENDIX

APPENDIX A: SEVERITY DEFINITIONS

Severity	Definition
Critical	This issue is straightforward to exploit and is likely to lead to catastrophic impact for client's reputation and can lead to financial loss for client or users.
High	This issue is difficult to exploit and is likely to lead to catastrophic impact for client's reputation and can lead to financial loss for client or users.
Medium	This issue is important to fix and puts a subset of users' data at risk and is possible to lead to moderate financial impact.
Low	This issue is not exploitable in a recurring basis and cannot have a significant impact on execution.
Note	This issue does not pose an immediate risk but is relevant to security best practices.

APPENDIX B: FILES IN SCOPE

```
src/Multisig.sol
src/State.sol
src/TokenRegistry.sol
src/Types.sol
```