Reya Network Security Review

Pashov Audit Group

Conducted by: T1MOH, Dan, merlinboii, ZanyBonzy

October 25th - September 30th

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **Reya-Labs/reya-network** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Reya Network

Reya Network is a trading-optimised modular L2 for perpetuals. The chain layer is powered by Arbitrum Orbit and is gas-free, with transactions ordered on a FIFO basis. The protocol layer directly tackles the vertical integration of DeFi applications by breaking the chain into modular components to support trading, such as PnL settlements, margin requirements, liquidations.

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - fb521866f00eb2fd7021a763a12aaf7d727e83f0

fixes review commit hash - 634058385163d38f1da033daf941c6fbf94884c6

Scope

The following smart contracts were in scope of the audit:

- `IAccountModule`
- `ICollateralPoolModule`
- `AccountExposure`
- `AccountModule`
- `CollateralPoolModule`
- `CollateralPool`
- `Market`
- `IDepositsModule`
- `IWithdrawalsModule`
- `Deposits`
- `Withdrawals`
- `DepositsModule`
- `WithdrawalsModule`
- `IAutoRebalanceModule`
- `IConfigurationModule`
- `ISharesModule`
- `DataTypes`
- `Errors`
- `Events`
- `FeatureFlagSupport`
- `AutoRebalanceModule`
- `ConfigurationModule`
- `SharesModule`
- `AllocationConfiguration`
- `GlobalConfiguration`
- `Pool`
- `ShareBalances`

7. Executive Summary

Over the course of the security review, T1MOH, Dan, merlinboii, ZanyBonzy engaged with Reya Network to review Reya Network. In this period of time a total of **2** issues were uncovered.

Protocol Summary

Protocol Name	Reya Network
Repository	https://github.com/Reya-Labs/reya-network
Date	October 25th - September 30th
Protocol Type	Perpetuals Trading L2

Findings Count

Severity	**Amount**
Critical	1
Medium	1
Total Findings	**2**

Summary of Findings

ID	Title	Severity	Status
[C-01]	Pool.removeLiquidityV2() uses incorrect token to send	Critical	Resolved
[M-01]	removeLiquidityBySigV2 does not correctly hash its contents to comply with EIP-712	Medium	Resolved

8. Findings

8.1. Critical Findings

[C-01] `Pool.removeLiquidityV2()` uses incorrect token to send

Severity

Impact: High

Likelihood: High

Description

`Pool.sol` will contain rUSD as quoteToken and deUSD, sdeUSD as supporting collaterals. The update introduces v2 versions of deposit and withdraw functions. It allows the deposit/withdraw of any following tokens: rUSD, deUSD, sdeUSD.

The problem is that by mistake `Pool.removeLiquidityV2()` always transfers quoteToken instead of withdrawing token. As a result, deUSD and sdeUSD cannot be withdrawn.

Recommendations

```
function removeLiquidityV2(
        Data storage self,
        address owner,
        RemoveLiquidityV2Input memory input
    )
        internal
        returns (uint256)
    {
        ...

        // withdraw from the core to the passive pool
        coreWithdrawal(self.accountId, input.token, tokenAmount);

-       // transfer quote token amount to the receiver
+       // transfer collateral token amount to the receiver
        // note, tokens are transferred to the receiver rather than the owner!
-       self.quoteToken.safeTransfer(input.receiver, tokenAmount);
+       input.token.safeTransfer(input.receiver, tokenAmount);

        return tokenAmount;
    }
```

8.2. Medium Findings

[M-01] `removeLiquidityBySigV2` does not correctly hash its contents to comply with EIP-712

Severity

Impact: Medium

Likelihood: Medium

Description

`removeLiquidityBySigV2` hashes the signature as shown below but doesn't fully hash it to comply with EIP-712.

```
Signature.validateRecoveredAddress(
            Signature.calculateDigest(
                keccak256(
                    abi.encode(
                        REMOVE_LIQUIDITY_V2_TYPEHASH,
                        block.chainid,
                        msg.sender,
                        owner,
                        poolId,
>>>                     abi.encode(
                            REMOVE_LIQUIDITY_V2_INPUT_TYPEHASH,
                            input.token,
                            input.sharesAmount,
                            input.receiver,
                            input.minOut
                        ),
                        Signature.incrementSigNonce(owner),
                        sig.deadline,
                        keccak256(extraSignatureData)
                    )
                )
            ),
            owner,
            sig
        );
```

The `RemoveLiquidityV2Input` struct is only encoded, not hashed as required by the standard.

> The struct values are encoded recursively as hashStruct(value).

As a result, EIP-compliant signers will have issues when attempting to use the `removeLiquidityBySigV2` function.

Recommendations

Hash the contents of the `RemoveLiquidityV2Input` struct.

```
Signature.validateRecoveredAddress(
            Signature.calculateDigest(
                keccak256(
                    abi.encode(
                        REMOVE_LIQUIDITY_V2_TYPEHASH,
                        block.chainid,
                        msg.sender,
                        owner,
                        poolId,
+                       keccak256(
                           abi.encode(
                              REMOVE_LIQUIDITY_V2_INPUT_TYPEHASH,
                              input.token,
                              input.sharesAmount,
                              input.receiver,
                              input.minOut
                          )
+                       ),
                        Signature.incrementSigNonce(owner),
                        sig.deadline,
                        keccak256(extraSignatureData)
                    )
                )
            ),
            owner,
            sig
        );
```