ZKFair Smart Contract
Audit Report

Thu Dec 21 2023

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

ZKFair Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	ZKFair is the first ZK-Rollup on ethereum based on Polygon CDK and Celestia DA
Type	Staking
Auditors	ScaleBit
Timeline	Thu Dec 21 2023 - Thu Dec 21 2023
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/ZKFair/zkfair-staking-contracts https://github.com/ZKFair/zkfair-transaction-mining-contract
Commits	https://github.com/ZKFair/zkfair-staking-contracts: 917a48b7a66494ce7d08ee9acd651eff018f4eb8 https://github.com/ZKFair/zkfair-transaction-mining-contract: c9c657a26b91db80ab41846faf7257a29ad68c75

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ZKFRC	contracts/ZKFRewardContract.sol	7284b54ae0860b84a1456c305ebc04bd020e2949
ZKFS	contracts/ZKFStaking.sol	6fbacdc70a04f4b4ebb454dda95e69fa5d552234
RDI	contracts/RewardDistribution.sol	5f483e2723c797a377f2ff6f6b34c34c54f30499

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	0	5
Informational	1	0	1
Minor	2	0	2
Medium	2	0	2
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by ZKFair-Smart-Contract to identify any potential issues and vulnerabilities in the source code of the ZKFair Smart Contract smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 5 issues of varying severity, listed below.

ID	Title	Severity	Status
RDI-1	Single-step Admin Transfer Can be Dangerous	Medium	Acknowledged
RDI-2	Lack of Events Emit	Minor	Acknowledged
ZKF-1	Incompatible with Deflation Tokens	Medium	Acknowledged
ZKF-2	Unnecessary Check	Informational	Acknowledged
ZKF1-1	Use `abi.encode` instead of `abi.encodePacked`	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the ZKFair Smart Contract Smart Contract:

Proposal Authority

- `proposalAuthority` has the power to propose a new Merkle root for the contract using the `proposerMerkleRoot` function.

Review Authority

- The `reviewAuthority` is tasked with approving or rejecting the proposed Merkle root. By using the `reviewPendingMerkleRoot` function, this role can update the `merkleRoot` and `rootUpdatedAt`.

Reward Sponsor

- The `rewardSponsor` can fund the contract by sending Ether directly. Their role is essential for maintaining the contract's balance and enabling reward distributions. They also have the authority to update their own address using `setRewardSponsor`.

User

- Users can invoke the `deposit` function to deposit tokens to contract.
- Users can invoke the `withdraw` function to withdraw tokens from contract if the duration time is up.
- Users can use the `claimReward` and `claim` functions to get the reward.

4 Findings

RDI-1 Single-step Admin Transfer Can be Dangerous

Severity: Medium

Status: Acknowledged

Code Location:

contracts/RewardDistribution.sol#70-78

Descriptions:

The `RewardDistribution.sol` contract assigns significant control to the `proposalAuthority` and `reviewAuthority` roles, which are critical for the integrity of the reward distribution process.

However, the functions `setProposalAuthority()` and `setReviewAuthority()` allow the current authorities to unilaterally transfer their roles to any address without a safeguarded transfer mechanism. This design can lead to accidental or malicious transfer of these roles, potentially compromising the reward distribution system.

The same issue may apply to `ZKFRewardContract.sol` and the admin roles as well.

Suggestion:

Implement a two-step transfer process for `proposalAuthority` and `reviewAuthority` roles. The current authority should first nominate a new address (the "push" step), and then the nominated address must accept the role (the "pull" step). Additionally, consider implementing a separate function for these roles to relinquish their authority, if necessary, to prevent unintentional role abandonment or misuse.

RDI-2 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

contracts/RewardDistribution.sol#74-94

Descriptions:

The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for those sensitive functions.

ZKF-1 Incompatible with Deflation Tokens

Severity: Medium

Status: Acknowledged

Code Location:

contracts/ZKFStaking.sol#73-74

Descriptions:

The `ZKFStaking.sol` contracts do not appear to support rebasing/deflationary/inflationary tokens whose balance changes during transfers or over time.

Suggestion:

It is recommended to add the necessary checks including at least verifying the amount of tokens transferred to contracts before and after the actual transfer to infer any fees/interest.

ZKF-2 Unnecessary Check

Severity: Informational

Status: Acknowledged

Code Location:

contracts/ZKFStaking.sol#120

Descriptions:

Since the variables in the `withdraw` function are obtained from the address corresponding to `msg.sender`, checking whether `depositInfo.depositor` is `msg.sender` is not necessary.

Suggestion:

It is suggested to remove unnecessary `if` check statements.

ZKF1-1 Use `abi.encode` instead of `abi.encodePacked`

Severity: Minor

Status: Acknowledged

Code Location:

contracts/ZKFRewardContract.sol#77

Descriptions:

Use `abi.encode()` instead which will pad items to 32 bytes, which will prevent hash collisions (e.g. `abi.encodePacked(0x123,0x456)` => `0x123456` => `abi.encodePacked(0x1,0x23456)`, but `abi.encode(0x123,0x456)` => `0x0...1230...456`). Unless there is a compelling reason, `abi.encode` should be preferred.

Suggestion:

It is recommended to use `abi.encode` as preferred.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.

- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.

- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.

- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.

- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.

- **Partially Fixed:** The issue has been partially resolved.

- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.