Kaito II

Audit Report

Thu Feb 13 2025

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

ScaleBit

Kaito II Audit Report

1 Executive Summary

1.1 Project Information

Description	This is an ERC4626 staking project.
Type	Staking
Auditors	ScaleBit
Timeline	Sat Feb 08 2025 - Tue Feb 11 2025
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/OpenKaito/kaito-contracts
Commits	925ebcebd54c9346c8fb1276db4523e730995ef1 b8d5b0e074c46a1e7850b77ea6c5447be7fbafe2 7e3d2b08971866d2dd202c3be6246050ce8851d6

1.2 Files in Scope

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

ID	File	SHA-1 Hash
SAAC	contracts/SingleAdminAccessControl.sol	c246c33a0ea601e7c29fb672de45798b1e59f7b4
KAI	contracts/Kaito.sol	b564e8d78884ebd85a4ccbf8caf89fae461dd28a
ISAAC	contracts/interfaces/ISingleAdminAccessControl.sol	6eef58c62f1d4cd846cd29c50b64c9d282f92189
ISKAITO	contracts/interfaces/IStakedKAITO.sol	fb9b3e6fd683fa61cd72857a3b0aaa1dfdcba671
SKAITO	contracts/StakedKAITO.sol	2540401c737194eb527296f8c3fc75cb544a59e7
APO	contracts/AgingPool.sol	fb703221e581dd0aa1b05643e4cf1c1d9552b7bc
ISKAITO	contracts/interfaces/IStakedKAITO.sol	16af4440893dc726e5c8f02460b0774f5b2d7f0e
SKAITO	contracts/StakedKAITO.sol	9727b641b6f79f6d6f971692e48a5eab473e237c

1.3 Issue Statistic

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

1.4 ScaleBit Audit Breakdown

ScaleBit 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
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

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) 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 Kaito to identify any potential issues and vulnerabilities in the source code of the Kaito II 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 2 issues of varying severity, listed below.

ID	Title	Severity	Status
APO-1	No Check for `transfer()` Return Value	Minor	Fixed
SKA-1	Centralization Risk	Major	Acknowledged

3 Participant Process

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

Admin

- Admin can update the address of owner through `transferAdmin()` and `acceptAdmin()` function.
- Admin can set the cooldown period of up to 90 days through the `setCooldownDuration()` function.
- Admin can set the vesting period of up to 90 days through the `setVestingPeriod()` function.
- Admin can burn the full restricted user amount and mint to the other address through the `redistributeLockedAmount()` function.
- Admin can withdraw the tokens of non-staked assets in the contract to any address through the `rescueTokens()` function.

Deployer

- Deployer of `Kaito.sol` contract can mint tokens for any address when invoking `constructor()` function.

REWARDER_ROLE

- `REWARDER_ROLE` can transfer reward tokens from the rewarder's account to the `StakedKAITO.sol` contract through the `transferInRewards()` function.

BLACKLIST_MANAGER_ROLE

- `BLACKLIST_MANAGER_ROLE` can add an address to the blacklist through the `addToBlacklist()` function.
- `BLACKLIST_MANAGER_ROLE` can remove an address from the blacklist through the `removeFromBlacklist()` function.

User

- Users can transfer tokens to another address through the `transfer()` function inherited from the `ERC20.sol` contract.
- User can approve other addresses to spend tokens on his behalf through the `approve()` function inherited from the `ERC20.sol` contract.
- User can increase the approval amount through the `increaseAllowance()` function inherited from the `ERC20.sol` contract.

- User can decrease the approval amount through the `decreaseAllowance()` function inherited from the `ERC20.sol` contract.
- User can transfer the token from an approved address to another address through the `transferFrom()` function inherited from the `ERC20.sol` contract.
- User can burn his tokens through the `burn()` function.
- User can use off-chain signatures to authorize other addresses to use his tokens through the `permit()` function inherited from the `ERC20Permit.sol` contract.
- User can withdraw a specified amount of underlying assets and destroy the corresponding shares through the `withdraw()` function.
- User can destroy a specified number of shares and withdraw the corresponding underlying assets through the `redeem()` function.
- User can deposit a specified amount of underlying assets and receive corresponding shares through the `deposit()` function inherited from the `ERC4626.sol` contract.
- User can mint a specified number of shares and deposit the corresponding underlying assets through the `mint()` function inherited from the `ERC4626.sol` contract.
- User can lock a specified amount of underlying assets during the cool-down period through the `cooldownAssets()` function.
- User can lock a specified number of shares during the cooldown period through the `cooldownShares()` function.
- User can withdraw their locked assets from the `AgingPool` after the cooling-off period ends through the `claimFromAP()` function.

4 Findings

APO-1 No Check for `transfer()` Return Value

Severity: Minor

Status: Fixed

Code Location:

contracts/AgingPool.sol#23

Descriptions:

While most tokens return true upon successful transfer and roll back transactions in case of transfer failure, there are still some tokens that return false when the transfer fails. If the return value is not checked, regardless of the success or failure of the transfer, it will be considered successful.

Suggestion:

It is recommended to add a return value check to the `transfer` operation.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

SKA-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/StakedKAITO.sol#56,73,84;

contracts/Kaito.sol#12

Descriptions:

Centralization risk was identified in the smart contract:

- The deployer of `Kaito.sol` can mint for any address.
- `BLACKLIST_MANAGER_ROLE` can add addresses to the blacklist to prevent users from staking and withdrawing and transferring.
- Admin can burn assets of blacklisted users and mint them for any address.
- Admin can withdraw the tokens of non-staked assets in the contract to any address through the `rescueTokens()` function.

Suggestion:

It is recommended that measures be taken to reduce the risk of centralization, such as a multi-signature mechanism.

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.