Cicada Finance
Audit Report

Fri Aug 23 2024

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

ScaleBit

Cicada Finance Audit Report

1 Executive Summary

1.1 Project Information

Description	Protocol Asset Management with a Equilibrium between Yield and Liquidity with RWA & Onchain assets. Real yield that are Secure, Scalable & Sustainable.
Type	Staking
Auditors	ScaleBit
Timeline	Mon Aug 19 2024 - Fri Aug 23 2024
Languages	Solidity
Platform	BTC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/mineral-devlop/cicada-contracts
Commits	639585f3e9f994fdd7cecd9548fba57e66d19af1 c9fee48e26d37a32bc0ed5b89165e2aad714cb95

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	5b59f8e0b269f625b97d65ede256c30c416b2983
SCB	contracts/StakeCiBtc.sol	28504004368ddecdaf85f91d80d796fe6657ffca
THE	contracts/utils/TransferHelper.sol	d3254b2420b833f7f41f253b57ffe504702cbb5b
MBT	contracts/MBtc.sol	f98087af27874db0cc9a3c2e1383758342237c06
SCB2	contracts/StakeCiBtc2.sol	1abe8ce4bbbd13077c44c0ca03c55babdfaa110c
CBT	contracts/CiBtc.sol	efb9ffc53683796c2689d73a991f160c90991587

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	9	5	4
Informational	1	0	1
Minor	3	2	1
Medium	2	1	1
Major	3	2	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 Cicada to identify any potential issues and vulnerabilities in the source code of the Cicada 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 9 issues of varying severity, listed below.

ID	Title	Severity	Status
CBT-1	Lack of Events Emit	Minor	Acknowledged
SCB-1	Centralization Risk	Major	Acknowledged
SCB-2	`deposits` Lack of Updates	Major	Fixed
SCB-3	Reentrancy Risk	Major	Fixed
SCB-4	`deposit` Sign Issue	Medium	Fixed
SCB-5	Incompatible With Deflationary Token	Medium	Acknowledged
SCB-6	Unnecessary Boolean Comparison	Minor	Fixed
SCB-7	Missing `msg.value` Number Limit	Minor	Fixed
SCB-8	Use `Calldata` Instead of `Memory` for Function Arguments That Do not Get Mutated	Informational	Acknowledged

3 Participant Process

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

Owner

- The owner can call the `transferAdmin` function to transfer Admin role to a address.
- The owner can call the `grantRole` to grant role for a address.
- The owner can call the `withdrawTokensSelf` function to withdraw all the token of the contract.

User

- Users can call the `deposit` function to stake the token by passing in the correct signature.
- Users can call the `withdraw` function to withdraw the token by passing in the correct signature.

4 Findings

CBT-1 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

contracts/CiBtc.sol#22-33

Descriptions:

The contract lacks appropriate events for monitoring `addMintRole()` , `removeMintRole()` , `mintTo()` , `burn()` operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for the those function.

SCB-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/StakeCiBtc.sol#81;

contracts/CiBtc.sol#30

Descriptions:

Centralization risk was identified in the smart contract.

- In the `CiBtc` contract, a user with `MINT_ROLE` privileges can `mint` tokens at will.
- User with `DEFAULT_ADMIN_ROLE` privileges can withdraw token from `StakeCiBtc` contract.

Suggestion:

It is recommended to take ways to reduce the risk of centralization.

SCB-2 `deposits` Lack of Updates

Severity: Major

Status: Fixed

Code Location:

contracts/StakeCiBtc.sol#168

Descriptions:

The signature identifies whether the `signature` has been used or not via the id field, but the function `deposit` lacks updates to mapping `deposits`, resulting in the same `signature` being used multiple times.

Suggestion:

It is recommended that mapping deposits be updated in time.

SCB-3 Reentrancy Risk

Severity: Major

Status: Fixed

Code Location:

contracts/StakeCiBtc.sol#185

Descriptions:

The `withdraw` function's lack of a `nonReentrant` modifier, along with the fact that the function do a call to `msg.sender`, results in a function that can be reentrant, allowing the same signature to be used multiple times.

Suggestion:

It is recommended to add `nonReentrant` modifier.

SCB-4 `deposit` Sign Issue

Severity: Medium

Status: Fixed

Code Location:

contracts/StakeCiBtc.sol#159

Descriptions:

The signature field of the `deposit` function does not contain the user's address, which could lead an attacker to listen for messages in the blockchain and obtain a valid signature to call the `deposit` function.

Suggestion:

It is recommended to ensure that this is in accordance with the design.

SCB-5 Incompatible With Deflationary Token

Severity: Medium

Status: Acknowledged

Code Location:

contracts/StakeCiBtc.sol#159-185

Descriptions:

In the `deposit` / `withdraw` function, due to the unknown address of the `token`, when the token is deflationary, the number of tokens transferred to the contract by the user may not be accurate.

Suggestion:

Since it's not known exactly what type of token this is, it's recommended to confirm whether such a question would conflict with the design philosophy.

SCB-6 Unnecessary Boolean Comparison

Severity: Minor

Status: Fixed

Code Location:

contracts/StakeCiBtc.sol#58

Descriptions:

There are statements in the contract that use Boolean variables to compare with Boolean values, such as `supportTokens[token] != true`, and it is recommended to just use that field's value directly.

Suggestion:

It is recommended to just use that field's value directly.

SCB-7 Missing `msg.value` Number Limit

Severity: Minor

Status: Fixed

Code Location:

contracts/StakeCiBtc.sol#159

Descriptions:

When a user call deposits with erc20 token, it is necessary to limit the `msg.value` to 0 to avoid loss of assets.

Suggestion:

It is recommended to limit the msg.value` value to 0 when token is erc20.

SCB-8 Use `Calldata` Instead of `Memory` for Function Arguments That Do not Get Mutated

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakeCiBtc.sol#66

Descriptions:

Mark data types as `calldata` instead of `memory` where possible. This makes it so that the data is not automatically loaded into `memory`. If the data passed into the function does not need to be changed (like updating values in an array), it can be passed in as `calldata`. The one exception to this is if the argument must later be passed into another function that takes an argument that specifies memory storage.

Suggestion:

It is recommended to use `calldata` instead of `memory`.

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