Camelot
Audit Report

Wed Apr 03 2024

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

ScaleBit

Camelot Audit Report

1 Executive Summary

1.1 Project Information

Description	Camelot is a Layer3 protocol which based on Merlin and dedicated to DePIN
Type	Blockchain & DeFI
Auditors	ScaleBit
Timeline	Tue Apr 02 2024 - Tue Apr 02 2024
Languages	Solidity
Platform	Merlin Chain
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/cam3lotAudit/contract https://github.com/cam3lot-pro/contract
Commits	https://github.com/cam3lotAudit/contract: c9fb831bcdb6c17c7cd52c64b969a92e41860750 https://github.com/cam3lot-pro/contract: 511fba95a8d4a08e76fc8da7669a44d335fa7f0a 86834e2ad4706bd5cbb5fbb3ff395f7ba5f2fb7d

1.2 Files in Scope

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

ID	File	SHA-1 Hash
SDM	contracts/lib/SafeDecimalMath.sol	8e39809be367e10f90d8630b18375b52942fbbdb
SMA	contracts/lib/SafeMath.sol	6a4b5e6e6895f3d4ea73b9122598eb388c206a5e
MST	contracts/MStake.sol	01f1c9125fd712e43de5f36d9fb58b14ae6e47f7

1.3 Issue Statistic

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

ID	Title	Severity	Status
MST-1	Lack of Events Emit	Minor	Acknowledged
MST-2	Unused Variable	Minor	Acknowledged
MST-3	Unused References	Minor	Acknowledged

3 Participant Process

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

Manager ROLE

- Manager can set `feePercent` to any value via `setFeePercent` function.
- Manager can set `feeAddress` to any address via `setFeeAddress` function.
- Manager can set `efficientDuration` to any value via the `setPriceEfficientDuration` function.
- Manager can set `canUnStake` to any value via `setCanUnStake` function.
- Manager can add any token address to the tokens supported by the contract via the `addAsset` function.
- Manager can remove tokens from the contract via the `removeAsset` function.

UPDATE PRICE ROLE

- UPDATE_PRICE_ROLE sets the price of each token to any value with the `updatePrice` function.

Users

- Users can stake ERC20 tokens into the contract by calling the `stake` function.
- Users can call the `stakeBTC` function to stake BTC into the contract.
- Users can call the `unstake` function to take out the ERC20 tokens pledged into a contract.
- Users can call the `unstakeBTC` function to take BTC out of the contract.

4 Findings

MST-1 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

contracts/MStake.sol#62-64;

contracts/MStake.sol#70-83;

contracts/MStake.sol#177-242

Descriptions:

The smart contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track important actions or detect potential issues. So it is recommended to add relevant events for some important functions like `setFeePercent()`, `stake()`, `unstake()`.

Suggestion:

It is recommended to emit events for those important functions.

MST-2 Unused Variable

Severity: Minor

Status: Acknowledged

Code Location:

contracts/MStake.sol#225;

contracts/MStake.sol#240

Descriptions:

The variable `data` returned by the call is not used in the contract and can be replaced by ``.

Suggestion:

It is recommended that `` be used instead of `bytes memory data`.

MST-3 Unused References

Severity: Minor

Status: Acknowledged

Code Location:

contracts/MStake.sol#5

Descriptions:

The protocol uses the Ownable contract of the openzeppelin library, but it is not used.

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
```

Suggestion:

It is recommended to delete useless references.

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.