MerlinStarter Staking Audit Report

Fri Apr 19 2024

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

ScaleBit

MerlinStarter Staking Audit Report

1 Executive Summary

1.1 Project Information

Description	The 1st native launchpad on MerlinLayer2
Type	Staking
Auditors	ScaleBit
Timeline	Tue Mar 26 2024 - Fri Apr 19 2024
Languages	Solidity
Platform	BTC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Merlinstarter/MerlinstarterStaking
Commits	817e4ce213ca68affa55375ee7e05aa0807fa60d b8c40a624ac9dbb1f04611dca5e450e203c3eb0d

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MSS	MerlinStarterStaking.sol	665f10d1a34e9d12ae6512252ea894bb549ea714

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	8	7	1
Informational	0	0	0
Minor	5	4	1
Medium	1	1	0
Major	2	2	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 MerlinStarter Staking to identify any potential issues and vulnerabilities in the source code of the MerlinStarter Staking 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 8 issues of varying severity, listed below.

ID	Title	Severity	Status
MSS-1	Centralization Risk	Major	Fixed
MSS-2	Incompatible With Deflationary Token	Major	Fixed
MSS-3	May not be able to Withdraw Funds	Medium	Fixed
VME-1	Lack of Events Emit	Minor	Acknowledged
VME-2	Lack of Zero Check in `mint`	Minor	Fixed
VME-3	Code Optimization	Minor	Fixed
VME-4	Unnecessary Assignments	Minor	Fixed
VME-5	Inaccurate Error Messages	Minor	Fixed

3 Participant Process

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

Admin

- Admin can change the parameters using the `setParameters` function.
- Admin can add white list account using the `addWhiteAccount` function.
- Admin can remove white list account using the `removeWhiteAccount` function.
- Admin can withdraw the token deposit in the contract to mFundAddress using the `withdrawToken` function.

User

- Users can stake tokens using the `stake` function.
- Users can unstake tokens and get rewards using the `exit` function.

4 Findings

MSS-1 Centralization Risk

Severity: Major

Status: Fixed

Code Location:

MerlinStarterStaking.sol#364-367

Descriptions:

This contract has centralization risk:

- Owner can call the `withdrawToken` function to take all the tokens stored in the staking contract.
- Owner can call the `mint` function to mint any number of tokens.

```
function withdrawToken(address tokenAddr,uint256 amount) external onlyOwner{
   IERC20 token = IERC20(tokenAddr);
   token.safeTransfer(_msgSender(), amount);
}
```

```
function mint(address _to,uint256 _amount) external returns (bool) {
   require(_Is_WhiteContractArr[msg.sender]," Only white address can mint !");
   _balances[_to] = _balances[_to].add(_amount);
   _totalSupply = _totalSupply.add(_amount);
   return true;
}
```

Suggestion:

It is recommended to implement decentralized governance mechanisms to distribute control and mitigate centralization risks. Specifically, consider implementing a multi-signature approval process for critical actions.

Resolution:

The client solves this problem by only withdrawing tokens to fundAddress.

MSS-2 Incompatible With Deflationary Token

Severity: Major

Status: Fixed

Code Location:

MerlinStarterStaking.sol#293

Descriptions:

In the `stake` function, due to the unknown address of `stakingToken`, when the token is deflationary, the amount 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.

Resolution:

The client has already solved the deflationary coin problem based on our advice.

MSS-3 May not be able to Withdraw Funds

Severity: Medium

Status: Fixed

Code Location:

MerlinStarterStaking.sol#293,330

Descriptions:

Since the pledged tokens are the same as the tokens that are rewarded, if no tokens are transferred in, this may result in subsequent users not being able to withdraw the pledged tokens from the pool.

Suggestion:

It is recommended to check if this is compatible with the design concept.

Resolution:

The client added the maximum number of pledges to solve this problem.

VME-1 Lack of Events Emit

Severity: Minor

Status: Acknowledged

Code Location:

VMErc20.sol#182,187;

MerlinStarterStaking.sol#335,339,343,349

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.

Resolution:

The client is already aware of the proposal.

VME-2 Lack of Zero Check in `mint`

Severity: Minor

Status: Fixed

Code Location:

VMErc20.sol#164-169

Descriptions:

When burning tokens, the tokens are transferred to address 0 and the total supply is reduced by an equal amount, whereas the `mint` function may mint tokens to address 0 and increase the total supply by an equal amount, which can lead to incorrect data logging. Possible impact on subsequent calculations.

Suggestion:

It's recommended to add checksums to disallow minting tokens to zero addresses.

Resolution:

The client has deleted the VMtoken contract.

VME-3 Code Optimization

Severity: Minor

Status: Fixed

Code Location:

VMErc20.sol#88-90;

MerlinStarterStaking.sol#162,163

Descriptions:

The value of a variable does not change after it is declared, the variable should be declared as `constant`.

Suggestion:

It is recommended to add constant to variables.

Resolution:

The client has resolved the issue.

VME-4 Unnecessary Assignments

Severity: Minor

Status: Fixed

Code Location:

VMErc20.sol#98

Descriptions:

In the `constructor`, the token for the number of `_totalSupply` is minted for the contract creator, but the initial `_totalSupply` itself is 0, and the uninitialized value in the mapping _balance was originally 0.

Suggestion:

It is recommended to confirm whether it is necessary to mint quantitative initial tokens for the contract creator, if not, you can delete the zero assignment.

Resolution:

The client has deleted the VMtoken contract.

VME-5 Inaccurate Error Messages

Severity: Minor

Status: Fixed

Code Location:

VMErc20.sol#149

Descriptions:

The `_transfer` function will report an error if from and to are not whitelisted addresses, but the error message is inaccurate and may mislead the user.

Suggestion:

It`s recommended to change the error message of require to be more accurate.

Resolution:

The client has refined the error message.

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