AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

TABLE OF CONTENTS

1. License
2. Disclaimer
3. Approach and methodology
4. Description
5. Findings

contact@auditmos.com

AUDITMOS

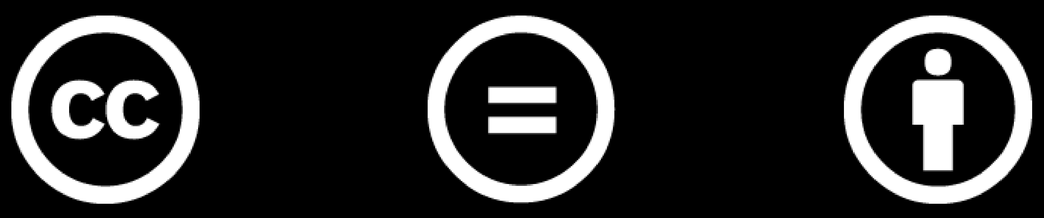
Secure your space

BITLOCUS

audit / code review report

January 10, 2022

LICENSE



contact@auditmos.com

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

DISCLAIMER

contact@auditmos.com

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

APPROACH AND METHODOLOGY

PURPOSE

1. Determine the correct operation of the protocol, according to the design specification.
2. Identify possible vulnerabilities that could be exploited by an attacker.
3. Detect errors in the smart contract that could lead to unexpected behavior.
4. Analyze whether best practices were followed during development.
5. Make recommendations to improve security and code readability.

CODEBASE

Repository	https://github.com/Bitlocus/token-lock
Branch	main
Commit hash	05b18d4ab56d9e137be8a65d7a843d815497b36c

METHODOLOGY

1. Reading the available documentation and understanding the code.
2. Doing automated code analysis and reviewing dependencies.
3. Checking manually source code line by line for security vulnerabilities.
4. Following guildlines and recommendations.
5. Preparing this report.

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

DESCRIPTION

Issues Categories:

Severity	Description
CRITICAL	vulnerability that can lead to loss of funds, failure to recover blocked funds, or catastrophic denial of service.
HIGH	vulnerability that can lead to incorrect contract state or unpredictable operation of the contract.
MEDIUM	failure to adhere to best practices, incorrect usage of primitives, without major impact on security.
LOW	recommendations or potential optimizations which can lead to better user experience or readability.

Each issue can be in the following state:

State	Description
PENDING	still waiting for resolving
ACKNOWLEDGED	know but not planned to resolve for some reasons
RESOLVED	fixed and deployed

contact@auditmos.com

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

FINDINGS

Finding	Severity	Status
#1 - Casting is not required in LockBase.sol	LOW	RESOLVED
#2 - Recommendations for gas optimisation	LOW	RESOLVED
#3 - Implement _addRecipient in more efficient way	LOW	RESOLVED
#4 - Modifier not required and typos	LOW	RESOLVED
#5 - Better business logic to avoid mistakes	LOW	ACKNOWLEDGED
#6 - Results from static code analisys	LOW	RESOLVED

contact@auditmos.com

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

#1 - CASTING IS NOT REQUIRED IN LOCKBASE.SOL

Severity	Status
LOW	RESOLVED

RECOMMENDATION

Variable _token is already an address, casting is not required.

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L44

contact@auditmos.com

BITLOCUS

audit / code review report

January 10, 2022

#2 - RECOMMENDATIONS FOR GAS OPTIMISATION

Severity	Status
LOW	RESOLVED

1) Function locked is used in recoverExcess function which consumes gas.

Gas optimisation may be good idea.

2) In _claim function use _claimed[recipient] instead of claimed(recipient) for gas optimisation

RECOMMENDATION

1) It would be more efficient to implement this function like below:

```
function locked() public view returns (uint256 amount) {
    uint256 recipientLen_ = _recipients.length;
    for (uint256 i = 0; i < recipientLen_; i++)
        amount += _allocation[_recipients[index]] - _claimed[_recipients[i]];
}
```

This example saves 4 function calls each loop.

After these 2 optimisation all 4 functions can became external - assuming that corresponding values are internal (not private) thus are usable in child contracts.

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L126

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L105

AUDITMOS

Secure your space

BITLOCUS

audit / code review report

January 10, 2022

#3 - IMPLEMENT _ADDRECIPIENT IN MORE EFFICIENT WAY

Severity	Status
LOW	RESOLVED

It seems that implemented function logic treats recipient as an address which has no zero allocation.

This cause additional iteration over recipients to find if particular one was already added.

RECOMMENDATION

Instead of:

```
for (uint256 i = 0; i < _recipients.length; i++)
    if (_recipients[i] == recipient)
        revert AlreadyRecipient(recipient);
```

Use:

```
if (_allocation[recipient] != 0)
 revert AlreadyRecipient(recipient);
```

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L66

contact@auditmos.com

BITLOCUS

audit / code review report

January 10, 2022

#4 - MODIFIER NOT REQUIRED AND TYPOS

Claim function doesn't need isRecipient(recipient) modifier as it is checked in _claim function.

Severity	Status
LOW	RESOLVED

TYPOS

provifed

- https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LinearLock.sol#L6

- https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/CliffLock.sol#L6

emopty

- https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/CliffLock.sol#L12

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L88

BITLOCUS

audit / code review report

January 10, 2022

#5 - BETTER BUSINESS LOGIC TO AVOID MISTAKES

1) Locker can every time remove recipient address from contract.

2) With recoverExcess admin can withdraw user funds, what will cause problems with withdrawals.

Severity	Status
LOW	ACKNOWLEDGED

RECOMMENDATION

Please consider below code and provide changes if needed:

function revoke(address recipient) external isRecipient(recipient)
onlyRole(LOCKER_ROLE)

and

function recoverExcess(IERC20 token_, address to, uint256 amount) external
onlyRole(LOCKER_ROLE)

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L117

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L124

BITLOCUS

audit / code review report

January 10, 2022

#6 - RESULTS FROM STATIC CODE ANALISYS

Severity	Status
LOW	RESOLVED

Reentrancy in LockBase.recoverExcess(IERC20,address,uint256):

External calls:

- token_.safeTransfer(to,amount)

Event emitted after the call(s):

- RecoverExcess(address(token_),to,amount)

It might be good idea to change an order or operations (safeTransfer as last statement of function call).

Reentrancy in LockBase._claim(address):

External calls:

- IERC20(token()).safeTransfer(recipient,amount)

Event emitted after the call(s):

- Claim(recipient,amount)

It might be good idea to change an order or operations (safeTransfer as last statement of function call).

PROOF OF SOURCE

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L129

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L130

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L93

https://github.com/Bitlocus/token-lock/blob/05b18d4ab56d9e137be8a65d7a843d815497b36c/contracts/LockBase.sol#L94

auditmos.com

Secure your space

contact@auditmos.com