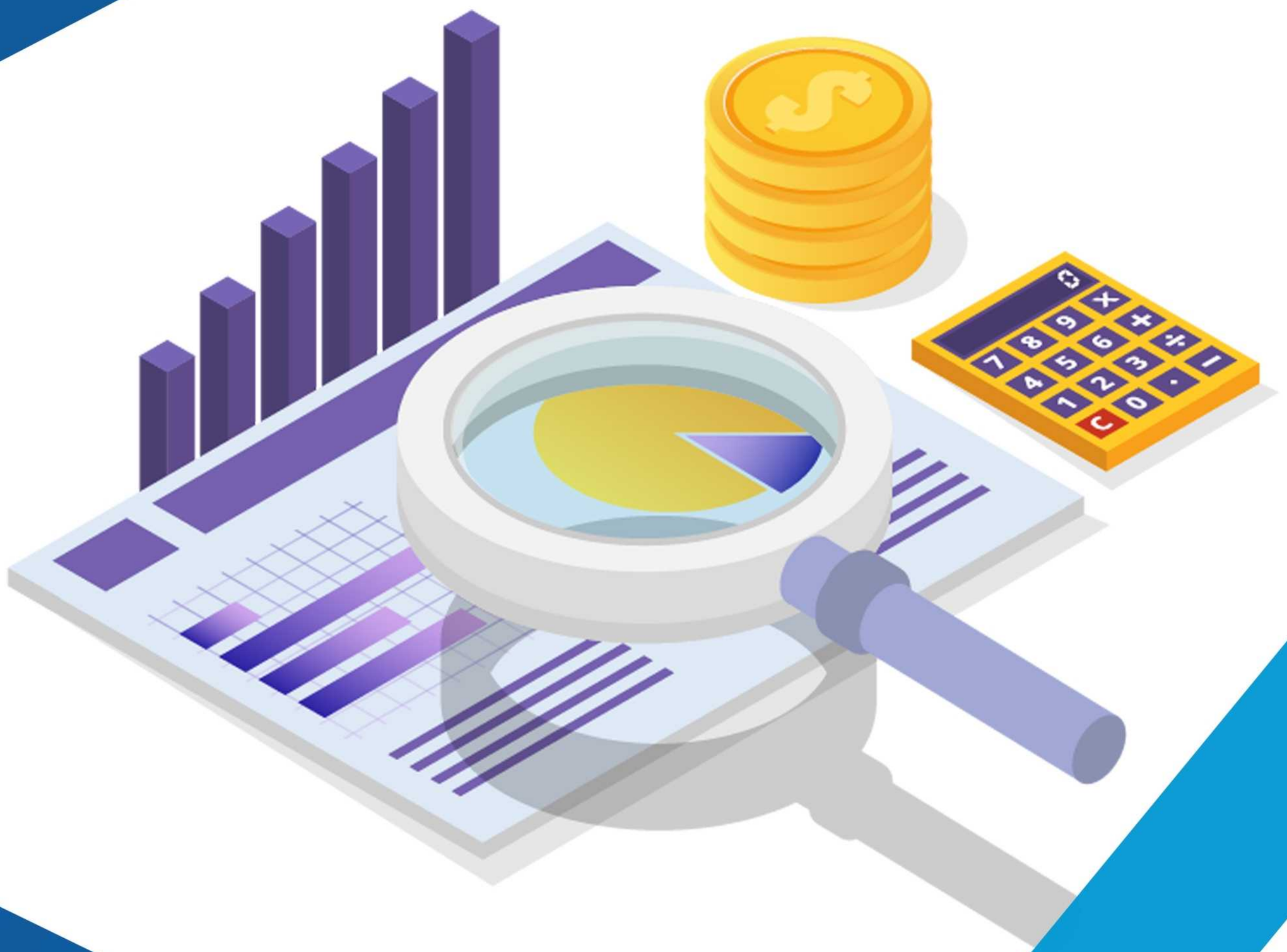
QuillAudits



Audit Report
June, 2021

locklet

Contents

Scope of Audit	01
Techniques and Methods	01
Issue Categories	02
Introduction	04
Issues Found – Code Review/Manual Testing	04
Automated Testing	06
Summary	09
Disclaimer	10

Scope of Audit

The scope of this audit was to analyze and document the LockletToken smart contract codebase for quality, security, and correctness.

Check Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	0	0	0
Closed	0	0	0	0

Introduction

During the period of **June 14, 2021 to June 15, 2021** - QuillAudits Team performed a security audit for **LockletToken** smart contracts.

The code for the audit was taken from following the official link:
https://github.com/locklet/locklet-evm-contracts/blob/main/contracts/LockletToken.sol
Commit hash: 28f037b

Issues Found – Code Review / Manual Testing

High severity issues

No Issues found.

Medium severity issues

No Issues found.

Low level severity issues

No Issues found.

Informational

No Issues found.

Functional test

Function Names	Testing results
transfer()	Passed
transferFrom()	Passed
increaseAllowance()	Passed
decreaseAllowance()	Passed
burn()	Passed
approve()	Passed

Automated Testing

Slither

```
INFO:Detectors:
LockletToken._totalSupply (LockletToken.sol#10) shadows:
        - ERC20._totalSupply (openzeppelin/contracts/token/ERC20/ERC20.sol#37)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
Context._msgData() (openzeppelin/contracts/utils/Context.sol#20-23) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (LockletToken.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/ERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.
7.6
Pragma version^0.8.0 (openzeppelin/contracts/token/ERC20/IERC20.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0
.7.6
Pragma version^0.8.0 (openzeppelin/contracts/utils/Context.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Redundant expression "this (openzeppelin/contracts/utils/Context.sol#21)" inContext (openzeppelin/contracts/utils/Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
LockletToken.constructor() (LockletToken.sol#12-16) uses literals with too many digits:
        - _initialSupply = 150000000 * 10 ** 18 (LockletToken.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (openzeppelin/contracts/token/ERC20/ERC20.sol#59-61)
symbol() should be declared external:
        - ERC20.symbol() (openzeppelin/contracts/token/ERC20/ERC20.sol#67-69)
decimals() should be declared external:
        - ERC20.decimals() (openzeppelin/contracts/token/ERC20/ERC20.sol#84-86)
totalSupply() should be declared external:
        - ERC20.totalSupply() (openzeppelin/contracts/token/ERC20/ERC20.sol#91-93)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (openzeppelin/contracts/token/ERC20/ERC20.sol#98-100)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#110-113)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (openzeppelin/contracts/token/ERC20/ERC20.sol#118-120)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#129-132)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#147-155)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#169-172)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (openzeppelin/contracts/token/ERC20/ERC20.sol#188-194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LockletToken.sol analyzed (4 contracts with 75 detectors), 20 result(s) found
```

Mythril

```
==== Potential denial-of-service if block gas limit is reached ====
SWC ID: 128
Severity: Low
Contract: 0x89B40A2d942BFC44a08071ab6C93d9DbDff4baFE
Function name: increaseAllowance(address,uint256)
PC address: 4752
Estimated Gas Usage: 1863 - 2718
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds t
he block gas limit.
--------------------
Initial State:

Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x14200000000000000, nonce:0, storage:{}

Transaction Sequence:

Caller: [SOMEGUY], function: approve(address,uint256), txdata: 0x095ea7b3000000000000000000000000101010101010101010101010101010101010101010101010
1010101010101010101010101010101010101010101010101, value: 0x0
Caller: [ATTACKER], function: increaseAllowance(address,uint256), txdata: 0x39509351000000000000000000000000101010101010101010101010101010101010101010
0101010101010101010101010101010101010101010101010101010101010101, value: 0x0

==== Potential denial-of-service if block gas limit is reached ====
SWC ID: 128
Severity: Low
Contract: 0x89B40A2d942BFC44a08071ab6C93d9DbDff4baFE
Function name: gasprice_bit_ether(int128)
PC address: 4838
Estimated Gas Usage: 1858 - 2333
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds t
he block gas limit.
--------------------
Initial State:

Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x4000000002000100, nonce:0, storage:{}

Transaction Sequence:

Caller: [SOMEGUY], function: approve(address,uint256), txdata: 0x095ea7b3000000000000000000000000101010101010101010101010101028002040101010101010
1010101010101010101010101010101010101010101010101, value: 0x0
Caller: [ATTACKER], function: gasprice_bit_ether(int128), txdata: 0x23b872dd000000000000000000000000101010101010101010101010101010101010101000000000000
0000000000001010101010101010101010101010101010101010000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

Theo

```
enderphan@enderphan contracts % theo --rpc-http HTTP://127.0.0.1:7545
The account's private key (input hidden)
>
Contract to interact with
> 0x89B40A2d942BFC44a08071ab6C93d9DbDff4baFE
Scanning for exploits in contract: 0x89B40A2d942BFC44a08071ab6C93d9DbDff4baFE
Connecting to HTTP: HTTP://127.0.0.1:7545.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file

Check the readme for more info:
https://github.com/cleanunicorn/theo

Theo version v0.8.2.
```

Solhint Linter

Gas & Economy

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 60:4:

Gas costs:

Gas requirement of function LockletToken.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 60:4:

Gas costs:

Gas requirement of function ERC20.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 68:4:

Gas costs:

Gas requirement of function LockletToken.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 68:4:

Gas costs:

Gas requirement of function ERC20.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 111:4:

Gas costs:

Gas requirement of function LockletToken.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 111:4:

Gas costs:

Gas requirement of function ERC20.allowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 119:4:

Gas costs:

Gas requirement of function LockletToken.allowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 119:4:

Solidity Static Analysis

contracts/LockletToken.sol:4:1: Error: Compiler version ^0.8.0 does not satisfy the r semver requirement

contracts/LockletToken.sol:12:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Integer Overflow and Underflow vulnerabilities, Reentrancy or Back-Door Entry were found in the contract.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the LockletToken platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the LockletToken Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

locklet

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com