EtherAuthority

www.EtherAuthority.io
audit@etherauthority.io

# SMART CONTRACT

## Security Audit Report

| | |
|---|---|
| Customer: | LoveCoin |
| Website: | lovecoinstaking.com |
| Platform: | Binance Smart Chain |
| Language: | Solidity |
| Date: | September 27th, 2021 |

# Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

# Introduction

EtherAuthority was contracted by the LoveCoin team to perform the Security audit of the LoveCoin and DiamondHearts Staking smart contracts code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on September 27th, 2021.

**The purpose of this audit was to address the following:**

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

# Project Background

DeFi Staking Contract – enables Lovecoin holders to stake their coins and earn interest paid in Lovecoin Tokens. The longer the user stakes, the more coins they can earn. Once a user stakes, they have to wait until their stake ends to claim their rewards and initial staking amount.

# Audit scope

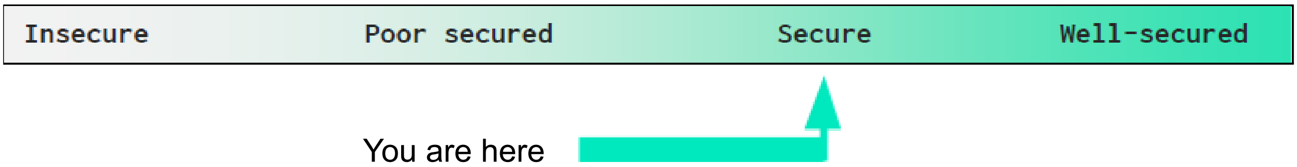
| **Name** | **Code Review and Security Analysis Report for LoveCoin and DiamondHearts Staking Contracts** |
|---|---|
| **Platform** | **BSC / Solidity** |
| **File 1** | DiamondHearts.sol |
| **File 1 MD5 Hash** | 077CEED4EAB80E177D1096A4FFB80CBC |
| **File 2** | LoveCoin.sol |
| **File 2 MD5 Hash** | 544AE3085F4C63272C7705867D26CC2C |
| **Audit Date** | September 27th, 2021 |

# Claimed Smart Contract Features

| Claimed Feature Detail | Our Observation |
|---|---|
| **File 1: LoveCoin.sol**<br>● Name: Lovecoin Token<br>● Symbol: Lovecoin<br>● Decimals: 8<br>● Half Life: 120 days<br>● Total Supply: 500 Trillion | **YES, This is valid.** |
| **File 2: DiamondHearts.sol**<br>**Staking Durations & Rewards**<br>● 6 Months – 1% Monthly Interest<br>● 1 Year – 2% Monthly Interest<br>● 3 Years – 3% Monthly Interest<br>● 5 Years – 4% Monthly Interest<br>● 10 Years – 5% Monthly Interest | **YES, This is valid.** |

# Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts contain owner control, which does not make it fully decentralized.



We used various tools like Slither, Solhint and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

**We found 0 critical, 0 high, 0 medium and 5 low and some very low level issues.**

**Investors Advice:** Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

# Technical Quick Stats

| Main Category | Subcategory | Result |
|---|---|---|
| Contract Programming | Solidity version not specified | Passed |
| | Solidity version too old | Moderated |
| | Integer overflow/underflow | Passed |
| | Function input parameters lack of check | Moderated |
| | Function input parameters check bypass | Passed |
| | Function access control lacks management | Passed |
| | Critical operation lacks event log | Moderated |
| | Human/contract checks bypass | Passed |
| | Random number generation/use vulnerability | Passed |
| | Fallback function misuse | Passed |
| | Race condition | Passed |
| | Logical vulnerability | Passed |
| | Features claimed | Passed |
| | Other programming issues | Moderated |
| Code Specification | Function visibility not explicitly declared | Passed |
| | Var. storage location not explicitly declared | Passed |
| | Use keywords/functions to be deprecated | Passed |
| | Unused code | Passed |
| Gas Optimization | “Out of Gas” Issue | Passed |
| | High consumption ‘for/while’ loop | Moderated |
| | High consumption ‘storage’ storage | Passed |
| | Assert() misuse | Passed |
| Business Risk | The maximum limit for mintage not set | Passed |
| | “Short Address” Attack | Passed |
| | “Double Spend” Attack | Passed |

**Overall Audit Result: PASSED**

## Code Quality

This audit scope has 2 smart contract files. Smart contracts also contain Libraries, Smart contracts, inherits and Interfaces.  These are compact and well written contracts.

The libraries in LoveCoin and DiamondHearts Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts.

The LoveCoin and DiamondHearts Protocol team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Code parts are **well** commented on smart contracts.

## Documentation

We were given a LoveCoin and DiamondHearts Staking smart contracts code in the form of the files. The hash of that code is mentioned above in the table.

As mentioned above, code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://lovecoinstaking.com which provided rich information about the project architecture and tokenomics.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries,  its functions are not used in external smart contract calls.

# AS-IS overview

## LoveCoin.sol

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | constructor | read | Passed | No Issue |
| 2 | totalSupply | read | Passed | No Issue |
| 3 | maxSupply | write | Passed | No Issue |
| 4 | decimals | read | Passed | No Issue |
| 5 | editMaxAirdrop | write | Function input parameters lack of check | Refer audit findings section below |
| 6 | editAdmin | write | Function input parameters lack of check | Refer audit findings section below |
| 7 | claimAdmin | write | Critical operation lacks event log | Refer audit findings section below |
| 8 | airdrop | write | Infinite loop | Refer audit findings section below |
| 9 | _airdrop | internal | Function input parameters lack of check | Refer audit findings section below |
| 10 | releaseCoins | write | Infinite loop | Refer audit findings section below |
| 11 | getOwner | external | Passed | No Issue |
| 12 | name | write | Passed | No Issue |
| 13 | symbol | write | Passed | No Issue |
| 14 | transfer | write | Passed | No Issue |
| 15 | balanceOf | read | Passed | No Issue |
| 16 | allowance | write | Passed | No Issue |
| 17 | approve | write | Passed | No Issue |
| 18 | transferFrom | write | Passed | No Issue |
| 19 | increaseAllowance | write | Passed | No Issue |
| 20 | decreaseAllowance | write | Passed | No Issue |
| 21 | mint | write | access only Owner | No Issue |
| 22 | _transfer | internal | Passed | No Issue |
| 23 | _mint | internal | Passed | No Issue |
| 24 | _burn | internal | Passed | No Issue |
| 25 | _approve | internal | Passed | No Issue |
| 26 | _burnFrom | internal | Passed | No Issue |

## DiamondHearts.sol

| Sl. | Functions | Type | Observation | Conclusion |
|---|---|---|---|---|
| 1 | constructor | read | Passed | No Issue |
| 2 | getStake | read | function return error | Refer audit findings section below |
| 3 | getStakeCount | read | Passed | No Issue |
| 4 | getVoteCounts | read | Passed | No Issue |
| 5 | getVotedStatus | read | Passed | No Issue |
| 6 | getRewardsPool | read | Passed | No Issue |
| 7 | vote | write | Infinite loop | Refer audit findings section below |
| 8 | createStake | write | Affiliate can be any address | No Issue |
| 9 | transferStake | write | Critical operation lacks event log | Refer audit findings section below |
| 10 | addToRewardsPool | write | Function input parameters lack of check | Refer audit findings section below |
| 11 | claimStake | write | Division before multiplication | Refer audit findings section below |
| 12 | withdrawAffiliateRewards | write | Critical operation lacks event log | Refer audit findings section below |
| 13 | editVotingReward | write | Passed | No Issue |
| 14 | editAdmin | write | Function input parameters lack of check | Refer audit findings section below |
| 15 | claimAdmin | write | Critical operation lacks event log | Refer audit findings section below |
| 16 | newBallot | write | Function input parameters lack of check | Refer audit findings section below |
| 17 | editInterestRates | write | Passed | No Issue |
| 18 | editAffiliateBonuses | write | Passed | No Issue |
| 19 | editReferredBonus | write | Passed | No Issue |
| 20 | _removeStakeFromList | internal | Passed | No Issue |

# Severity Definitions

| Risk Level | Description |
|---|---|
| Critical | Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc. |
| High | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial |
| Medium | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| Low | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| Lowest / Code Style / Best Practice | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |

# Audit Findings

## DiamondHearts.sol

### Critical

No Critical severity vulnerabilities were found.

### High

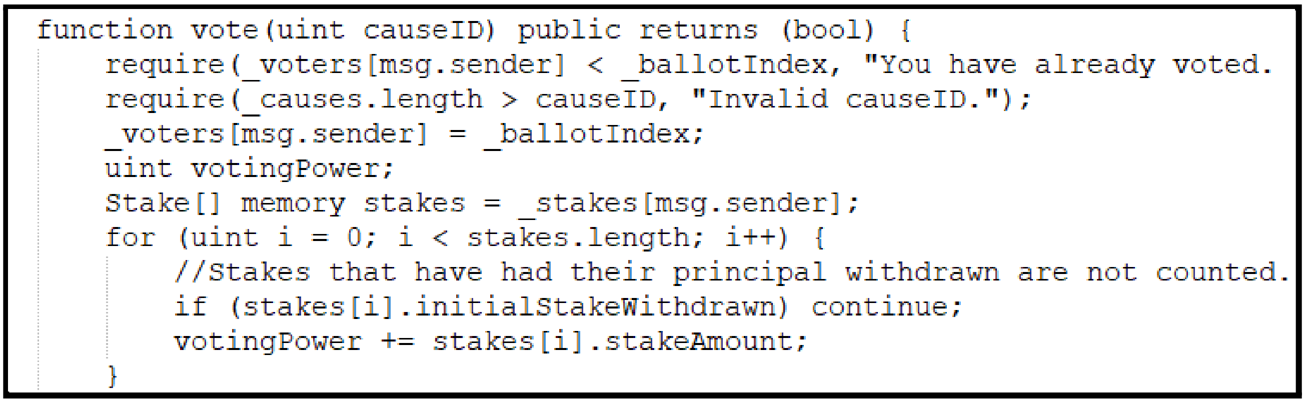
No High severity vulnerabilities were found.

### Medium

No Medium severity vulnerabilities were found.

### Low

(1) Infinite loop possibility:



```
function vote(uint causeID) public returns (bool) {
    require(_voters[msg.sender] < _ballotIndex, "You have already voted.
    require(_causes.length > causeID, "Invalid causeID.");
    _voters[msg.sender] = _ballotIndex;
    uint votingPower;
    Stake[] memory stakes = _stakes[msg.sender];
    for (uint i = 0; i < stakes.length; i++) {
        //Stakes that have had their principal withdrawn are not counted.
        if (stakes[i].initialStakeWithdrawn) continue;
        votingPower += stakes[i].stakeAmount;
    }
```

The vote function has a “for loop”, which does not have a stakes length limit. This will cost more gas as stake value will increase.

**Resolution**: Upper limit should be limited in the “for loops”.

(2) Function input parameters lack of check:

Variable validation is not performed in below functions:

- addToRewardsPool = amount
- editAdmin = newAdmin
- newBallot = numOfCauses.

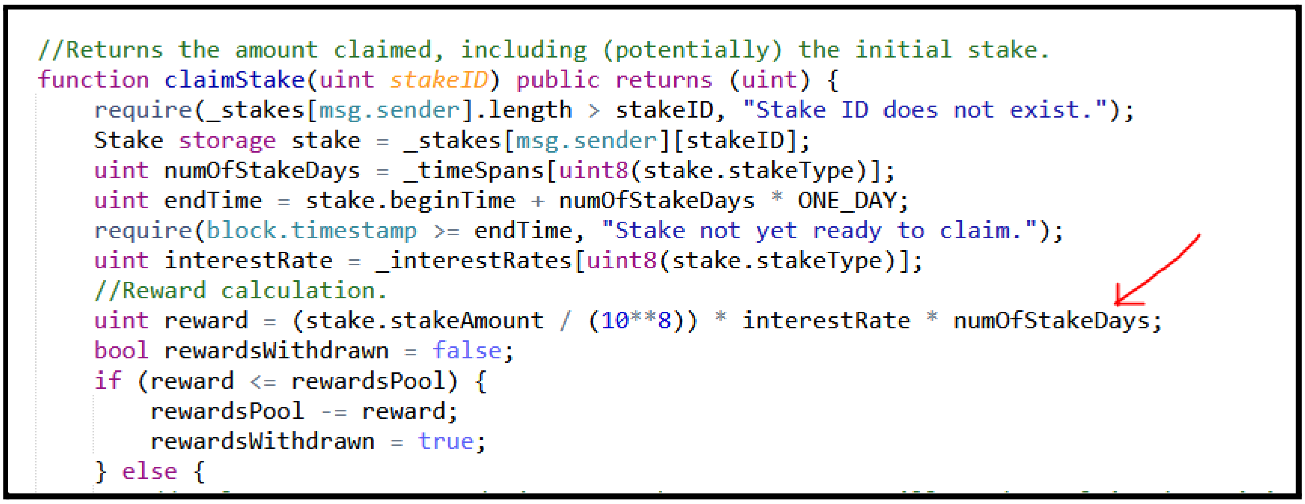
**Resolution**: We suggest having conditions which check for input params for proper expected values, to prevent any data discrepancy.

(3) Critical operation lacks event log:

It is recommended to fire appropriate events for the following functions which change their state significantly. This is helpful to be coordinated by the clients.

- withdrawAffiliateRewards
- addToRewardsPool
- transferStake
- vote.

(4) Division before multiplication:



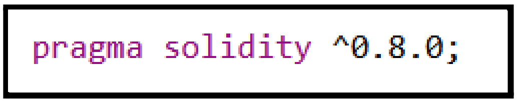
```
//Returns the amount claimed, including (potentially) the initial stake.
function claimStake(uint stakeID) public returns (uint) {
    require(_stakes[msg.sender].length > stakeID, "Stake ID does not exist.");
    Stake storage stake = _stakes[msg.sender][stakeID];
    uint numOfStakeDays = _timeSpans[uint8(stake.stakeType)];
    uint endTime = stake.beginTime + numOfStakeDays * ONE_DAY;
    require(block.timestamp >= endTime, "Stake not yet ready to claim.");
    uint interestRate = _interestRates[uint8(stake.stakeType)];
    //Reward calculation.
    uint reward = (stake.stakeAmount / (10**8)) * interestRate * numOfStakeDays;
    bool rewardsWithdrawn = false;
    if (reward <= rewardsPool) {
        rewardsPool -= reward;
        rewardsWithdrawn = true;
    } else {
```

Solidity being resource constrained language, dividing any amount and then multiplying will cause discrepancies in the outcome. Therefore, always multiply the amount first and then divide it.

**Resolution**: Consider ordering multiplication before division.

## Very Low / Informational / Best practices:

(1) Use latest solidity version:



```
pragma solidity ^0.8.0;
```

Using the latest solidity will prevent any compiler level bugs.

**Resolution**: Please use 0.8.7 which is the latest version.

## LoveCoin.sol

### Critical

No Critical severity vulnerabilities were found.

### High

No High severity vulnerabilities were found.

### Medium

No Medium severity vulnerabilities were found.

### Low

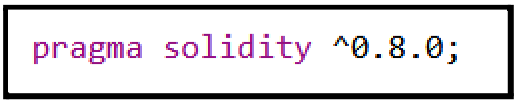
(1) High gas consuming loops:

In airdrop() and releaseCoins() functions, there are “for loops”, which do not have array (addresses and periodsToRelease) length limits. This will cost more gas when the array length increases.

**Resolution**: Upper limit should be limited in for loops.

### Very Low / Informational / Best practices:

(1) Use latest solidity version:



```
pragma solidity ^0.8.0;
```

Using the latest solidity will prevent any compiler level bugs.

**Resolution**: Please use 0.8.7 which is the latest version.

# Conclusion

We were given  contract codes. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are not critical ones. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high-level description of functionality was presented in the As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

# Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

**Manual Code Review:**

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

**Vulnerability Analysis:**

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

### Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

### Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

# Disclaimers

## EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

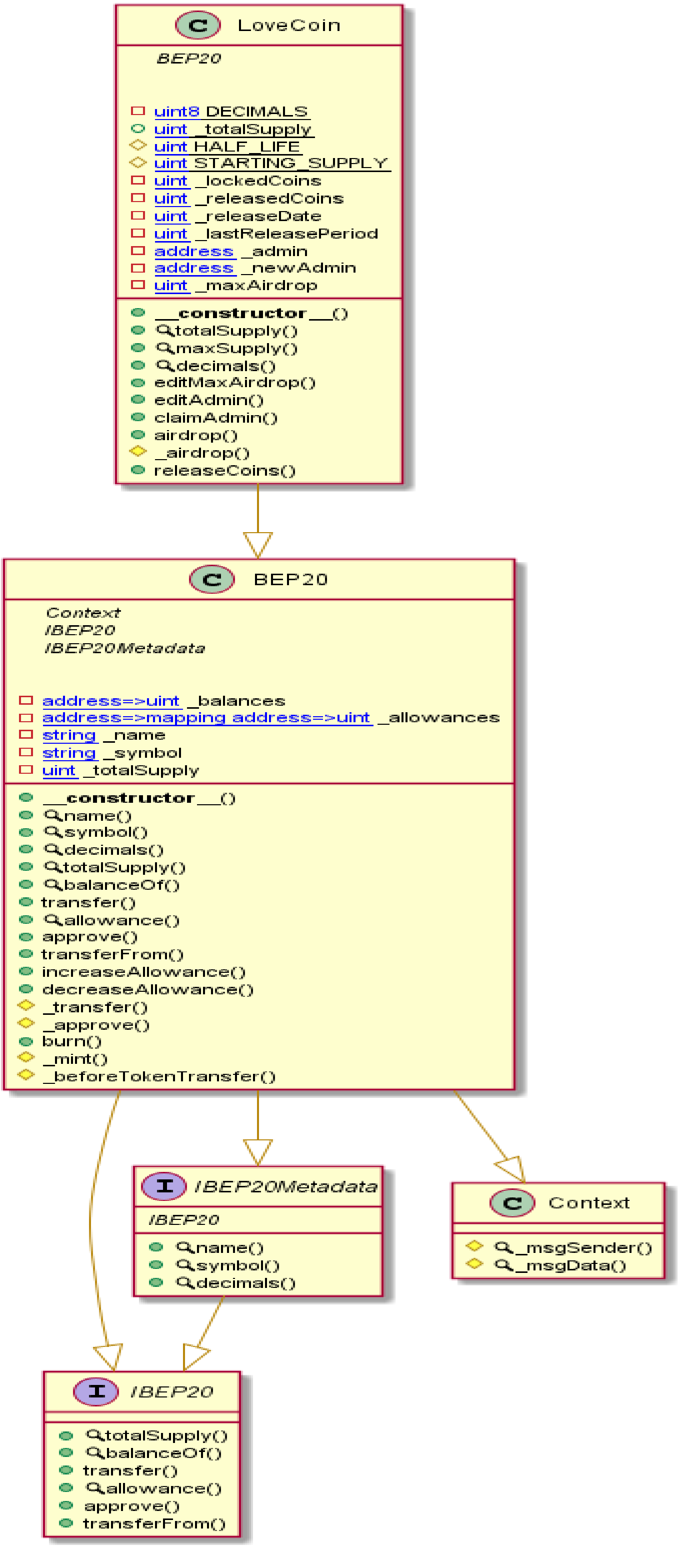
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

## Technical Disclaimer

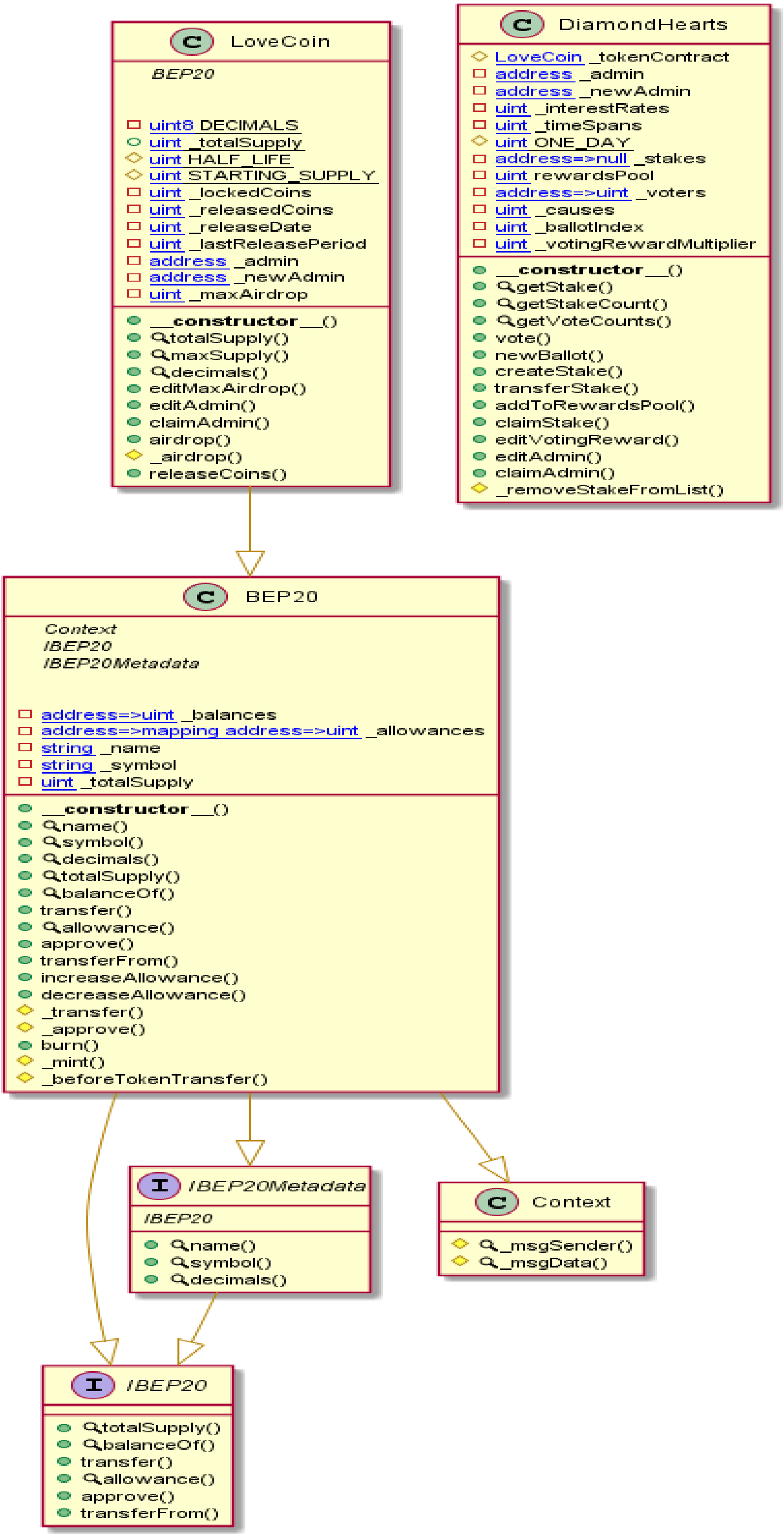
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

# Appendix

## Code Flow Diagram
## LoveCoin Token

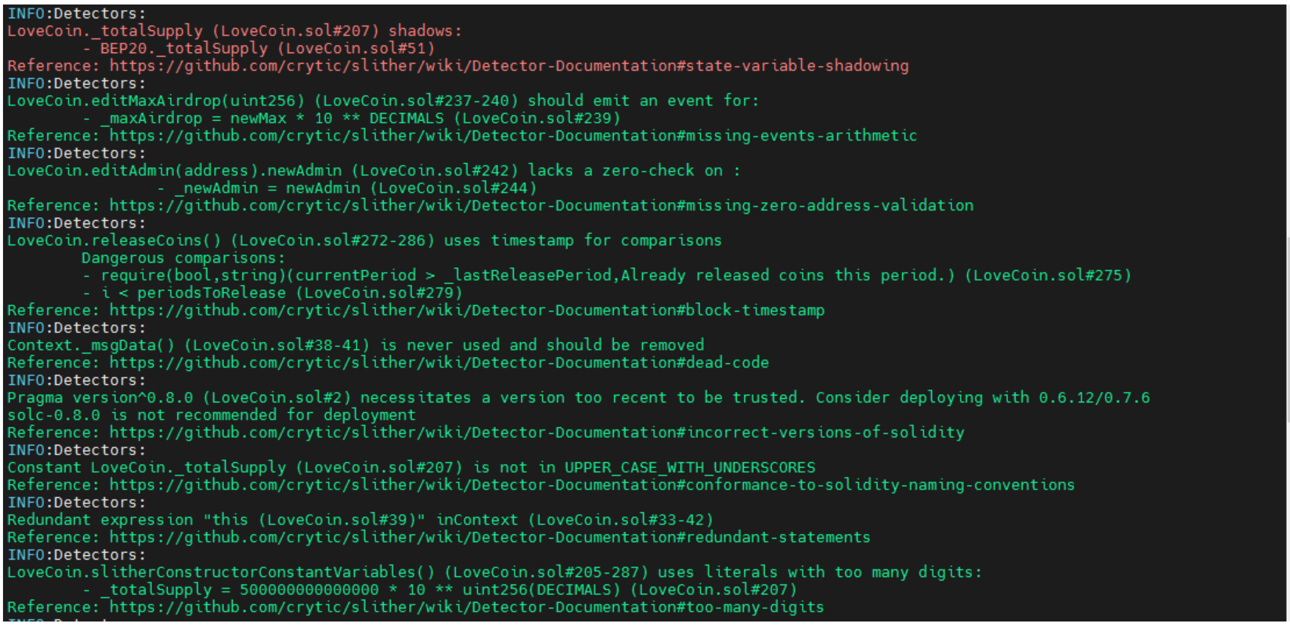


# Diamond Hearts Token

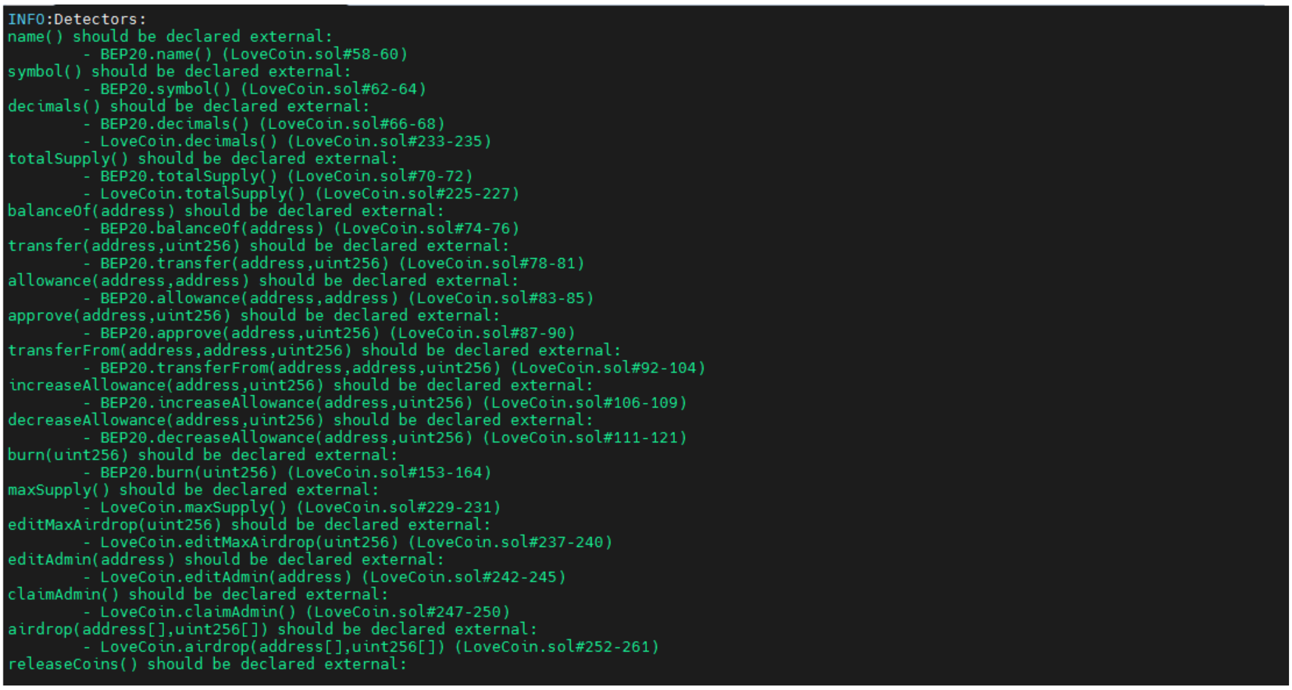


# Slither Results Log

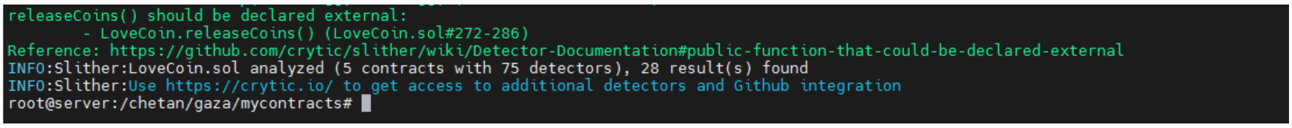
## Slither log >> LoveCoin.sol



```
INFO:Detectors:
LoveCoin._totalSupply (LoveCoin.sol#207) shadows:
        - BEP20._totalSupply (LoveCoin.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
LoveCoin.editMaxAirdrop(uint256) (LoveCoin.sol#237-240) should emit an event for:
        - _maxAirdrop = newMax * 10 ** DECIMALS (LoveCoin.sol#239)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
LoveCoin.editAdmin(address).newAdmin (LoveCoin.sol#242) lacks a zero-check on :
                - _newAdmin = newAdmin (LoveCoin.sol#244)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
LoveCoin.releaseCoins() (LoveCoin.sol#272-286) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(currentPeriod > _lastReleasePeriod,Already released coins this period.) (LoveCoin.sol#275)
        - i < periodsToRelease (LoveCoin.sol#279)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (LoveCoin.sol#38-41) 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 (LoveCoin.sol#2) 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:
Constant LoveCoin._totalSupply (LoveCoin.sol#207) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (LoveCoin.sol#39)" inContext (LoveCoin.sol#33-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
LoveCoin.slitherConstructorConstantVariables() (LoveCoin.sol#205-287) uses literals with too many digits:
        - _totalSupply = 500000000000000 * 10 ** uint256(DECIMALS) (LoveCoin.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
```

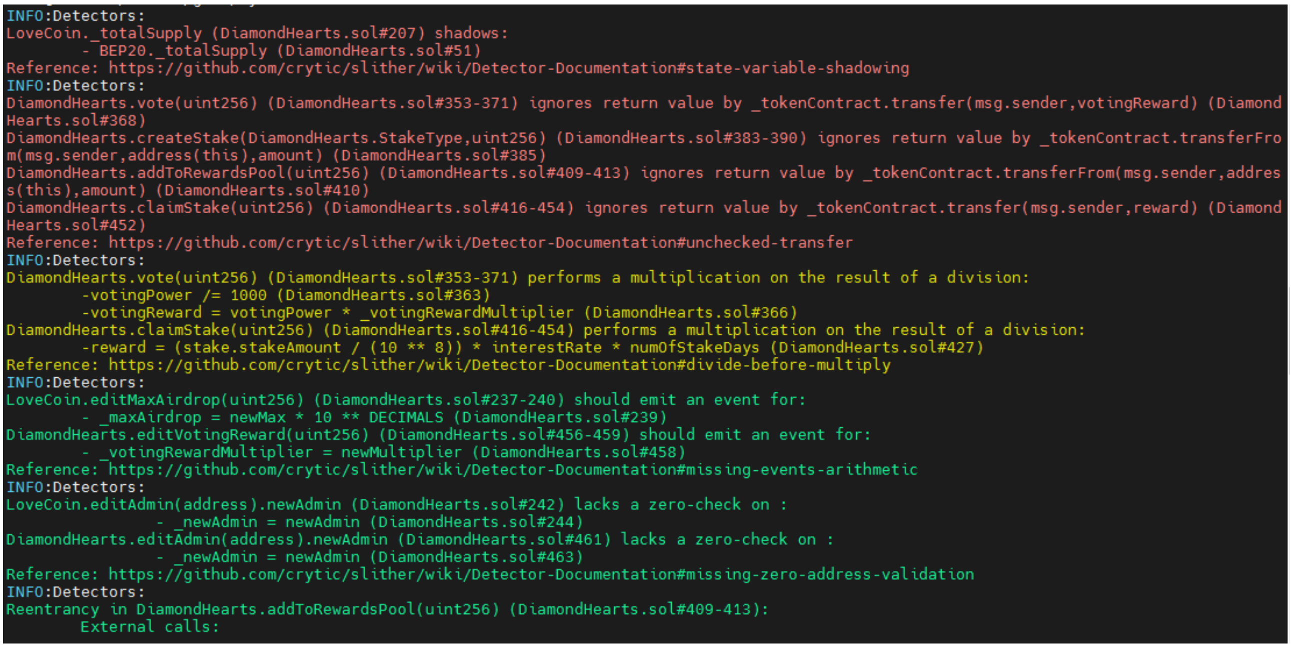


```
INFO:Detectors:
name() should be declared external:
        - BEP20.name() (LoveCoin.sol#58-60)
symbol() should be declared external:
        - BEP20.symbol() (LoveCoin.sol#62-64)
decimals() should be declared external:
        - BEP20.decimals() (LoveCoin.sol#66-68)
        - LoveCoin.decimals() (LoveCoin.sol#233-235)
totalSupply() should be declared external:
        - BEP20.totalSupply() (LoveCoin.sol#70-72)
        - LoveCoin.totalSupply() (LoveCoin.sol#225-227)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (LoveCoin.sol#74-76)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (LoveCoin.sol#78-81)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (LoveCoin.sol#83-85)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (LoveCoin.sol#87-90)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (LoveCoin.sol#92-104)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (LoveCoin.sol#106-109)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (LoveCoin.sol#111-121)
burn(uint256) should be declared external:
        - BEP20.burn(uint256) (LoveCoin.sol#153-164)
maxSupply() should be declared external:
        - LoveCoin.maxSupply() (LoveCoin.sol#229-231)
editMaxAirdrop(uint256) should be declared external:
        - LoveCoin.editMaxAirdrop(uint256) (LoveCoin.sol#237-240)
editAdmin(address) should be declared external:
        - LoveCoin.editAdmin(address) (LoveCoin.sol#242-245)
claimAdmin() should be declared external:
        - LoveCoin.claimAdmin() (LoveCoin.sol#247-250)
airdrop(address[],uint256[]) should be declared external:
        - LoveCoin.airdrop(address[],uint256[]) (LoveCoin.sol#252-261)
releaseCoins() should be declared external:
```

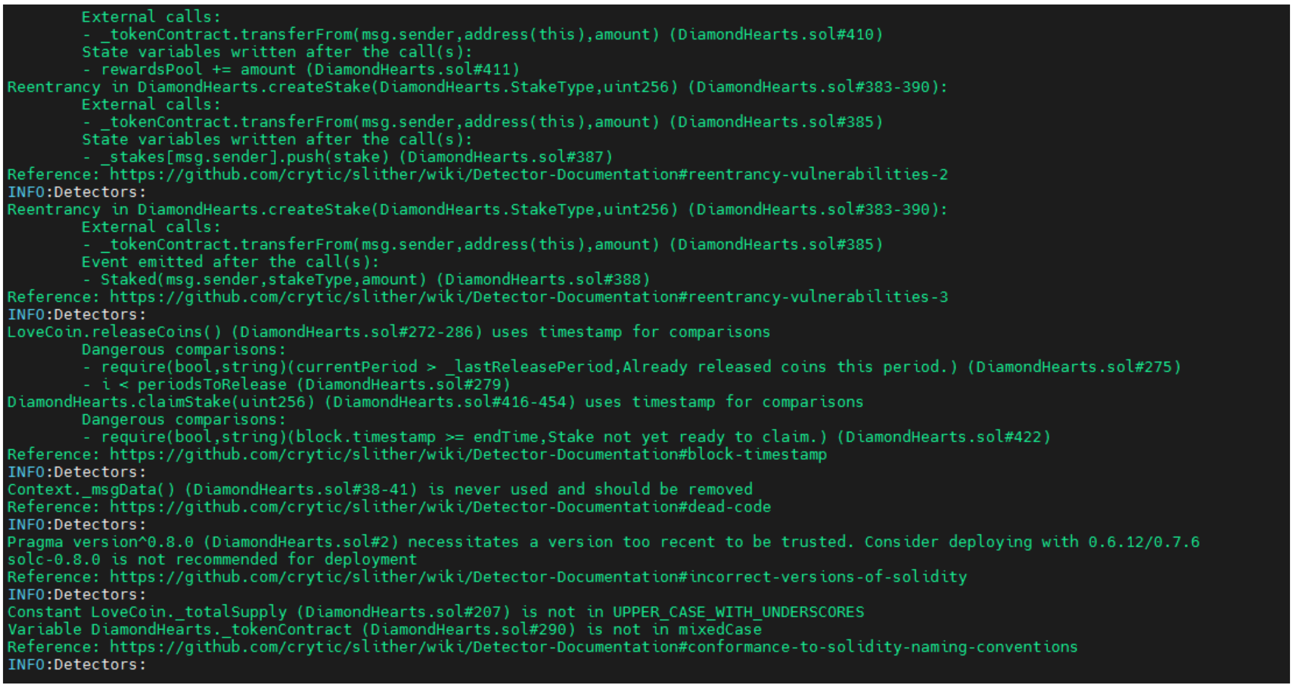


```
releaseCoins() should be declared external:
        - LoveCoin.releaseCoins() (LoveCoin.sol#272-286)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:LoveCoin.sol analyzed (5 contracts with 75 detectors), 28 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
root@server:/chetan/gaza/mycontracts#
```

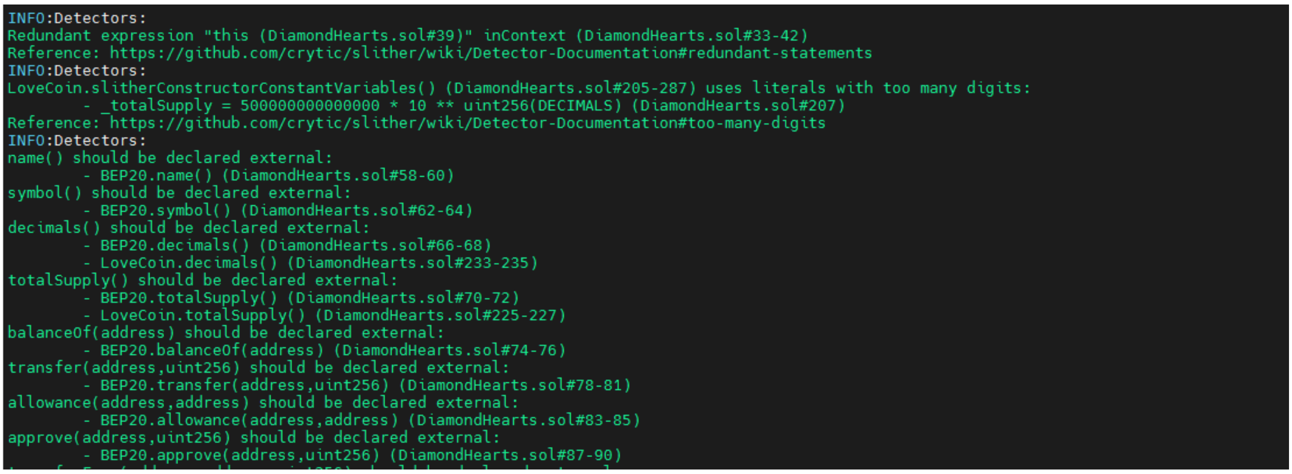
## Slither log >> DiamondHearts.sol



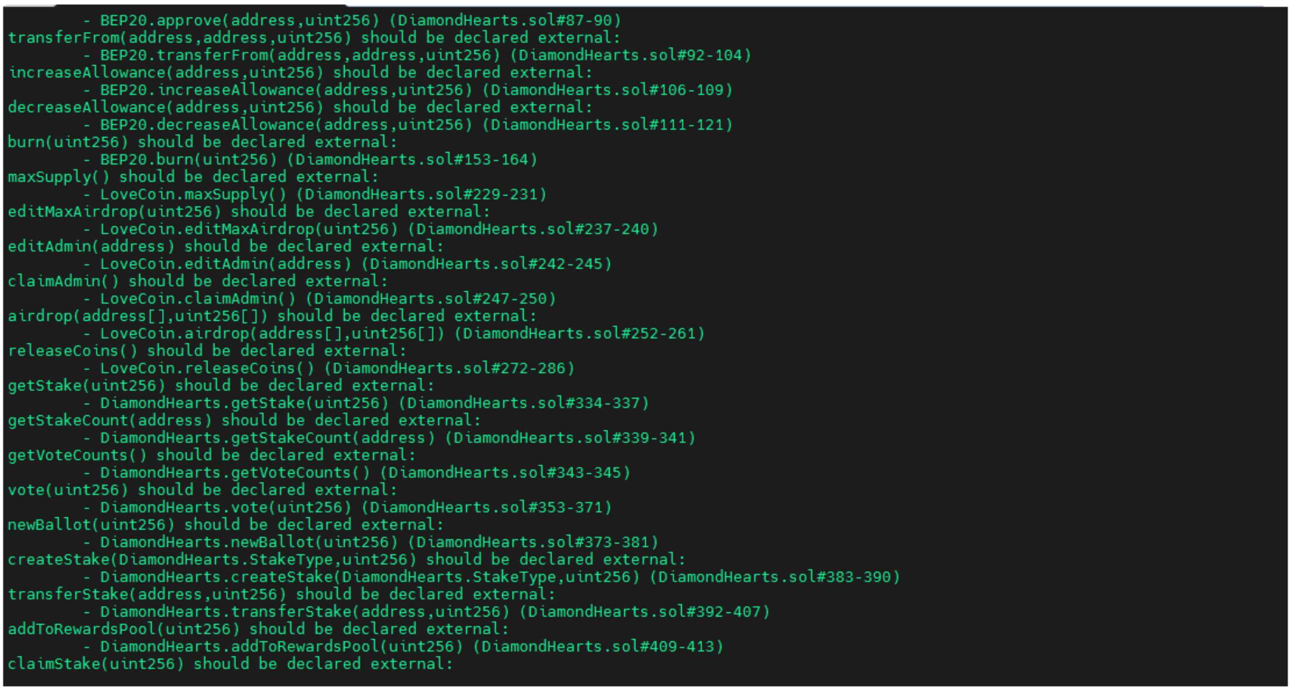
```
INFO:Detectors:
LoveCoin._totalSupply (DiamondHearts.sol#207) shadows:
        - BEP20._totalSupply (DiamondHearts.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
DiamondHearts.vote(uint256) (DiamondHearts.sol#353-371) ignores return value by _tokenContract.transfer(msg.sender,votingReward) (Diamond
Hearts.sol#368)
DiamondHearts.createStake(DiamondHearts.StakeType,uint256) (DiamondHearts.sol#383-390) ignores return value by _tokenContract.transferFro
m(msg.sender,address(this),amount) (DiamondHearts.sol#385)
DiamondHearts.addToRewardsPool(uint256) (DiamondHearts.sol#409-413) ignores return value by _tokenContract.transferFrom(msg.sender,addres
s(this),amount) (DiamondHearts.sol#410)
DiamondHearts.claimStake(uint256) (DiamondHearts.sol#416-454) ignores return value by _tokenContract.transfer(msg.sender,reward) (Diamond
Hearts.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
DiamondHearts.vote(uint256) (DiamondHearts.sol#353-371) performs a multiplication on the result of a division:
        -votingPower /= 1000 (DiamondHearts.sol#363)
        -votingReward = votingPower * _votingRewardMultiplier (DiamondHearts.sol#366)
DiamondHearts.claimStake(uint256) (DiamondHearts.sol#416-454) performs a multiplication on the result of a division:
        -reward = (stake.stakeAmount / (10 ** 8)) * interestRate * numOfStakeDays (DiamondHearts.sol#427)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
LoveCoin.editMaxAirdrop(uint256) (DiamondHearts.sol#237-240) should emit an event for:
        - _maxAirdrop = newMax * 10 ** DECIMALS (DiamondHearts.sol#239)
DiamondHearts.editVotingReward(uint256) (DiamondHearts.sol#456-459) should emit an event for:
        - _votingRewardMultiplier = newMultiplier (DiamondHearts.sol#458)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
LoveCoin.editAdmin(address).newAdmin (DiamondHearts.sol#242) lacks a zero-check on :
                - _newAdmin = newAdmin (DiamondHearts.sol#244)
DiamondHearts.editAdmin(address).newAdmin (DiamondHearts.sol#461) lacks a zero-check on :
                - _newAdmin = newAdmin (DiamondHearts.sol#463)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in DiamondHearts.addToRewardsPool(uint256) (DiamondHearts.sol#409-413):
        External calls:
```



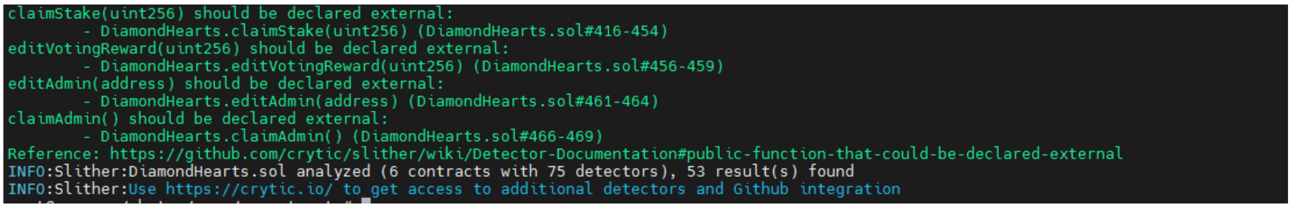
```
        External calls:
        - _tokenContract.transferFrom(msg.sender,address(this),amount) (DiamondHearts.sol#410)
        State variables written after the call(s):
        - rewardsPool += amount (DiamondHearts.sol#411)
Reentrancy in DiamondHearts.createStake(DiamondHearts.StakeType,uint256) (DiamondHearts.sol#383-390):
        External calls:
        - _tokenContract.transferFrom(msg.sender,address(this),amount) (DiamondHearts.sol#385)
        State variables written after the call(s):
        - _stakes[msg.sender].push(stake) (DiamondHearts.sol#387)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in DiamondHearts.createStake(DiamondHearts.StakeType,uint256) (DiamondHearts.sol#383-390):
        External calls:
        - _tokenContract.transferFrom(msg.sender,address(this),amount) (DiamondHearts.sol#385)
        Event emitted after the call(s):
        - Staked(msg.sender,stakeType,amount) (DiamondHearts.sol#388)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LoveCoin.releaseCoins() (DiamondHearts.sol#272-286) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(currentPeriod > _lastReleasePeriod,Already released coins this period.) (DiamondHearts.sol#275)
        - i < periodsToRelease (DiamondHearts.sol#279)
DiamondHearts.claimStake(uint256) (DiamondHearts.sol#416-454) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= endTime,Stake not yet ready to claim.) (DiamondHearts.sol#422)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Context._msgData() (DiamondHearts.sol#38-41) 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 (DiamondHearts.sol#2) 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:
Constant LoveCoin._totalSupply (DiamondHearts.sol#207) is not in UPPER_CASE_WITH_UNDERSCORES
Variable DiamondHearts._tokenContract (DiamondHearts.sol#290) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
```



```
INFO:Detectors:
Redundant expression "this (DiamondHearts.sol#39)" inContext (DiamondHearts.sol#33-42)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
LoveCoin.slitherConstructorConstantVariables() (DiamondHearts.sol#205-287) uses literals with too many digits:
        - _totalSupply = 500000000000000 * 10 ** uint256(DECIMALS) (DiamondHearts.sol#207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
name() should be declared external:
        - BEP20.name() (DiamondHearts.sol#58-60)
symbol() should be declared external:
        - BEP20.symbol() (DiamondHearts.sol#62-64)
decimals() should be declared external:
        - BEP20.decimals() (DiamondHearts.sol#66-68)
        - LoveCoin.decimals() (DiamondHearts.sol#233-235)
totalSupply() should be declared external:
        - BEP20.totalSupply() (DiamondHearts.sol#70-72)
        - LoveCoin.totalSupply() (DiamondHearts.sol#225-227)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (DiamondHearts.sol#74-76)
transfer(address,uint256) should be declared external:
        - BEP20.transfer(address,uint256) (DiamondHearts.sol#78-81)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (DiamondHearts.sol#83-85)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (DiamondHearts.sol#87-90)
```



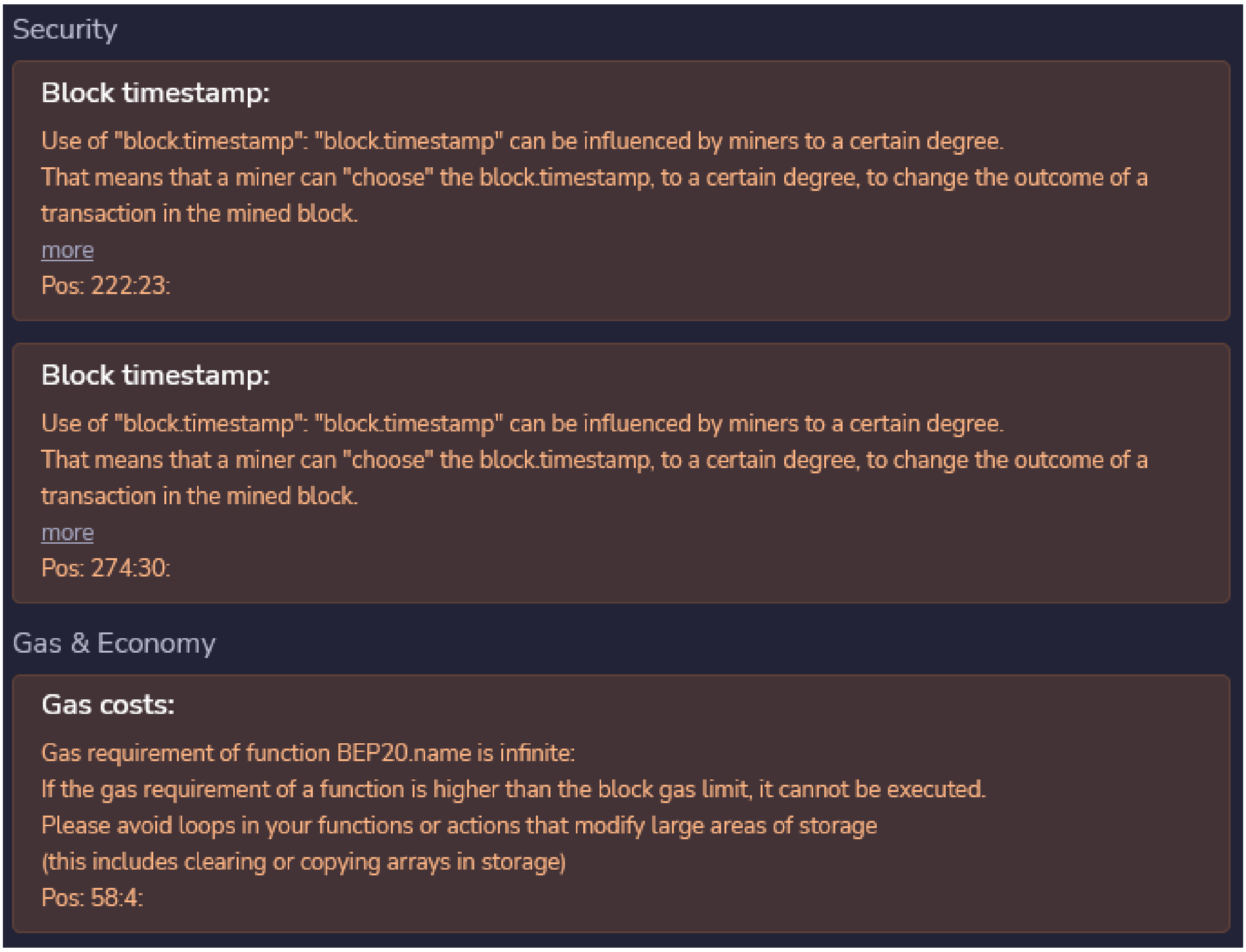
```
        - BEP20.approve(address,uint256) (DiamondHearts.sol#87-90)
transferFrom(address,address,uint256) should be declared external:
        - BEP20.transferFrom(address,address,uint256) (DiamondHearts.sol#92-104)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (DiamondHearts.sol#106-109)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (DiamondHearts.sol#111-121)
burn(uint256) should be declared external:
        - BEP20.burn(uint256) (DiamondHearts.sol#153-164)
maxSupply() should be declared external:
        - LoveCoin.maxSupply() (DiamondHearts.sol#229-231)
editMaxAirdrop(uint256) should be declared external:
        - LoveCoin.editMaxAirdrop(uint256) (DiamondHearts.sol#237-240)
editAdmin(address) should be declared external:
        - LoveCoin.editAdmin(address) (DiamondHearts.sol#242-245)
claimAdmin() should be declared external:
        - LoveCoin.claimAdmin() (DiamondHearts.sol#247-250)
airdrop(address[],uint256[]) should be declared external:
        - LoveCoin.airdrop(address[],uint256[]) (DiamondHearts.sol#252-261)
releaseCoins() should be declared external:
        - LoveCoin.releaseCoins() (DiamondHearts.sol#272-286)
getStake(uint256) should be declared external:
        - DiamondHearts.getStake(uint256) (DiamondHearts.sol#334-337)
getStakeCount(address) should be declared external:
        - DiamondHearts.getStakeCount(address) (DiamondHearts.sol#339-341)
getVoteCounts() should be declared external:
        - DiamondHearts.getVoteCounts() (DiamondHearts.sol#343-345)
vote(uint256) should be declared external:
        - DiamondHearts.vote(uint256) (DiamondHearts.sol#353-371)
newBallot(uint256) should be declared external:
        - DiamondHearts.newBallot(uint256) (DiamondHearts.sol#373-381)
createStake(DiamondHearts.StakeType,uint256) should be declared external:
        - DiamondHearts.createStake(DiamondHearts.StakeType,uint256) (DiamondHearts.sol#383-390)
transferStake(address,uint256) should be declared external:
        - DiamondHearts.transferStake(address,uint256) (DiamondHearts.sol#392-407)
addToRewardsPool(uint256) should be declared external:
        - DiamondHearts.addToRewardsPool(uint256) (DiamondHearts.sol#409-413)
claimStake(uint256) should be declared external:
```



```
claimStake(uint256) should be declared external:
        - DiamondHearts.claimStake(uint256) (DiamondHearts.sol#416-454)
editVotingReward(uint256) should be declared external:
        - DiamondHearts.editVotingReward(uint256) (DiamondHearts.sol#456-459)
editAdmin(address) should be declared external:
        - DiamondHearts.editAdmin(address) (DiamondHearts.sol#461-464)
claimAdmin() should be declared external:
        - DiamondHearts.claimAdmin() (DiamondHearts.sol#466-469)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:DiamondHearts.sol analyzed (6 contracts with 75 detectors), 53 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

# Solidity Static Analysis

### LoveCoin.sol



Security

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 222:23:

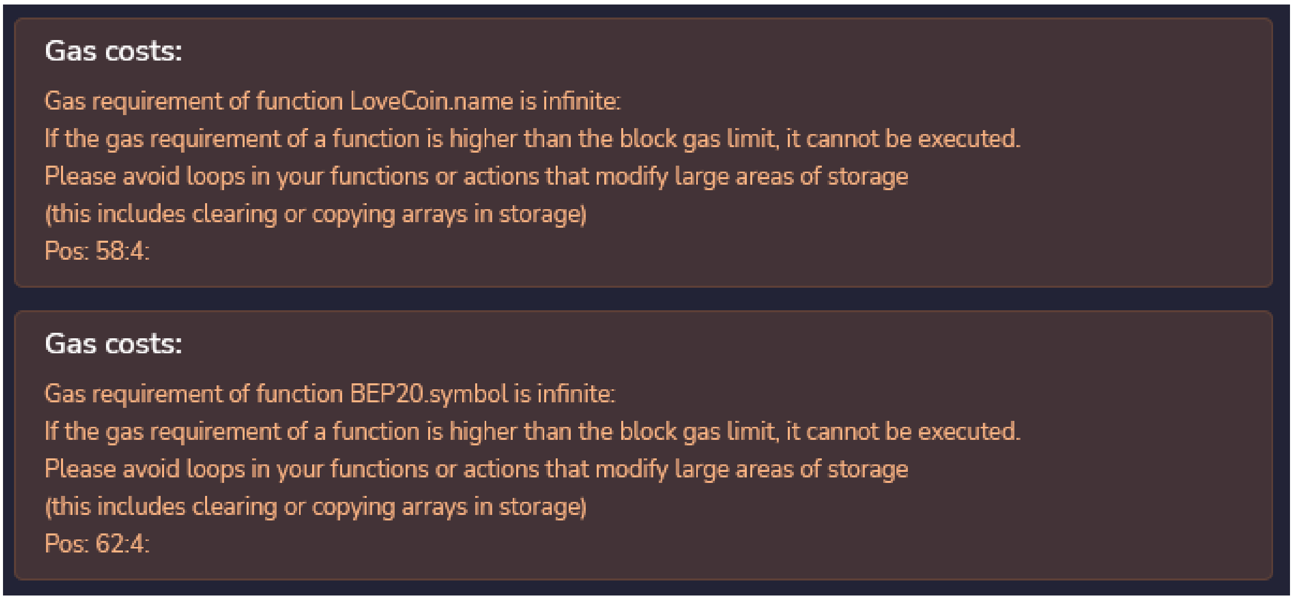
**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 274:30:

Gas & Economy

**Gas costs:**

Gas requirement of function BEP20.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: 58:4:

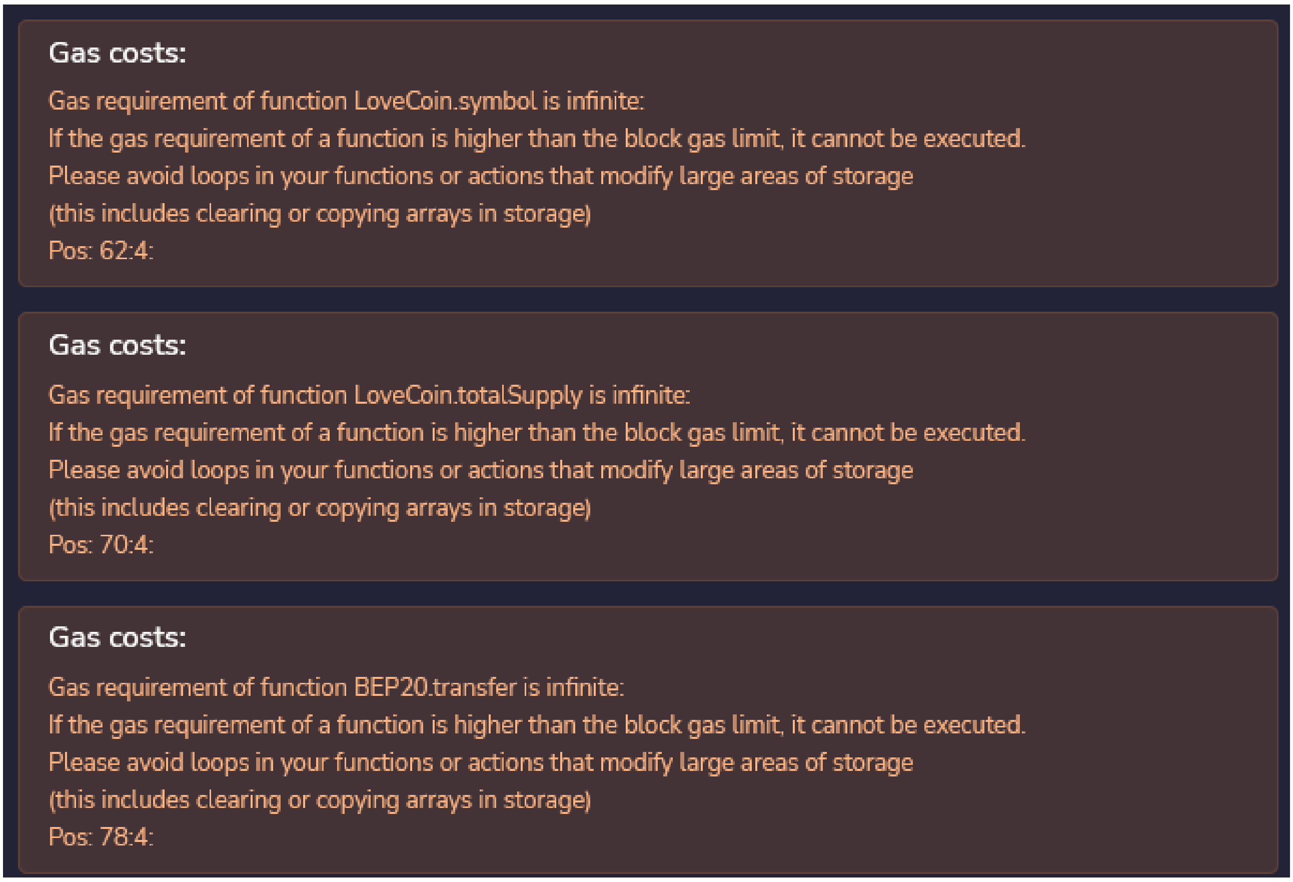


**Gas costs:**

Gas requirement of function LoveCoin.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: 58:4:

**Gas costs:**

Gas requirement of function BEP20.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: 62:4:



Gas costs:

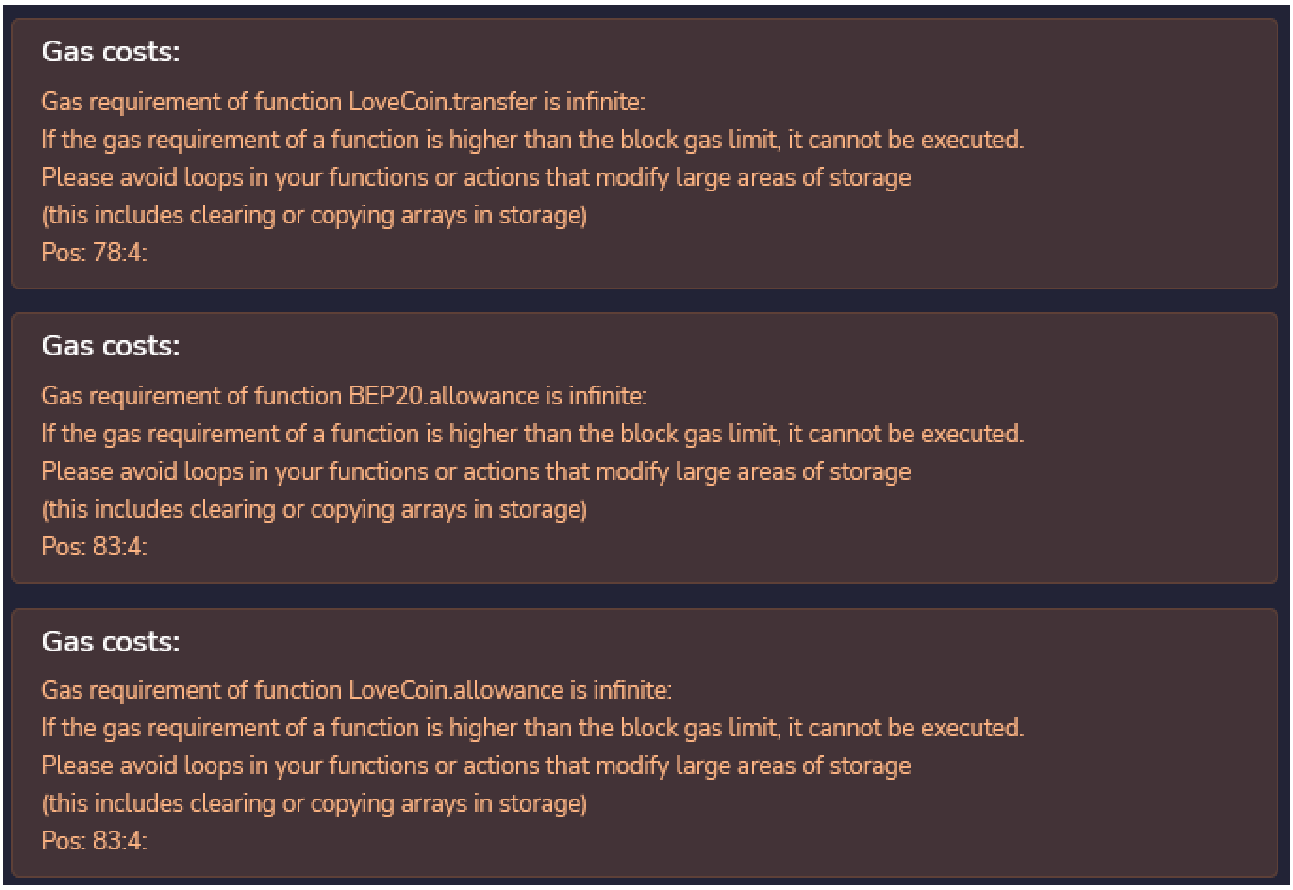
Gas requirement of function LoveCoin.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 62:4:

Gas costs:

Gas requirement of function LoveCoin.totalSupply 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: 70:4:

Gas costs:

Gas requirement of function BEP20.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: 78:4:



Gas costs:

Gas requirement of function LoveCoin.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: 78:4:

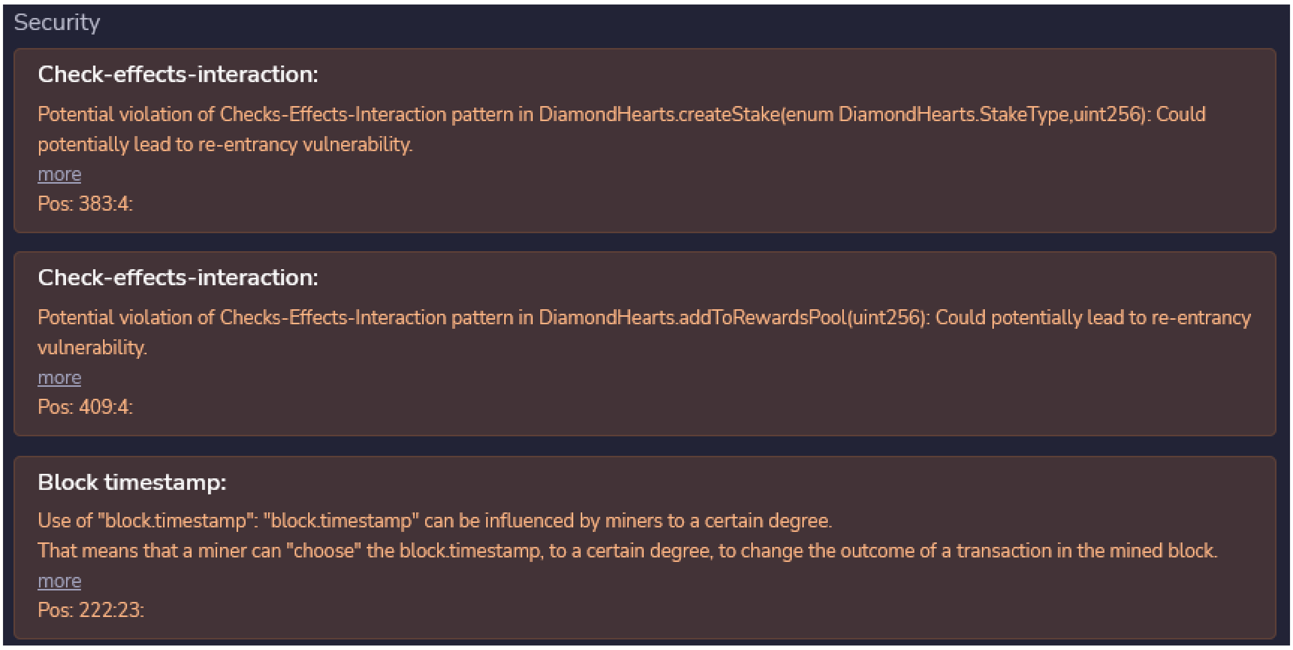
Gas costs:

Gas requirement of function BEP20.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: 83:4:

Gas costs:

Gas requirement of function LoveCoin.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: 83:4:

## DiamondHearts.sol



Security

**Check-effects-interaction:**

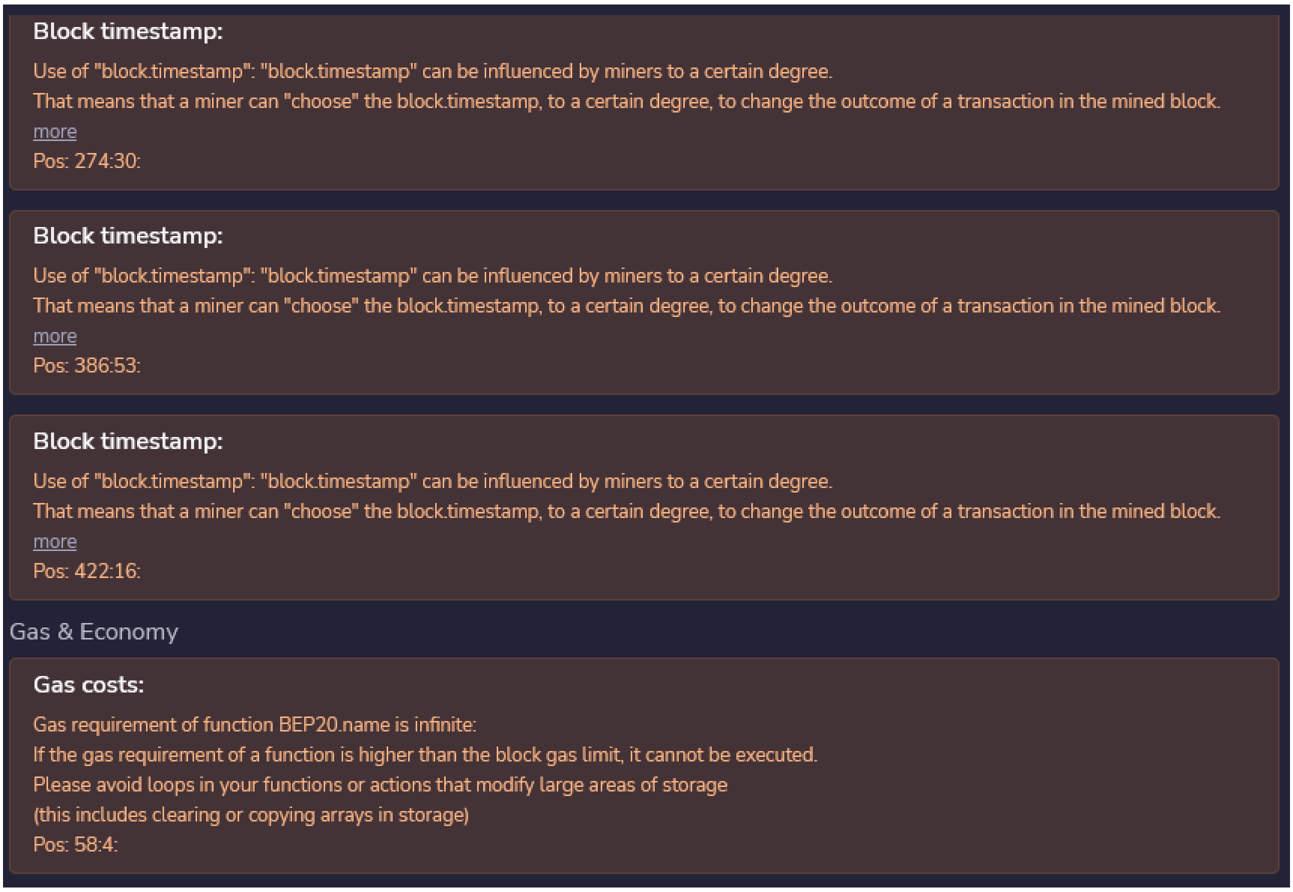
Potential violation of Checks-Effects-Interaction pattern in DiamondHearts.createStake(enum DiamondHearts.StakeType,uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 383:4:

**Check-effects-interaction:**

Potential violation of Checks-Effects-Interaction pattern in DiamondHearts.addToRewardsPool(uint256): Could potentially lead to re-entrancy vulnerability.
more
Pos: 409:4:

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 222:23:



**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 274:30:

**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 386:53:

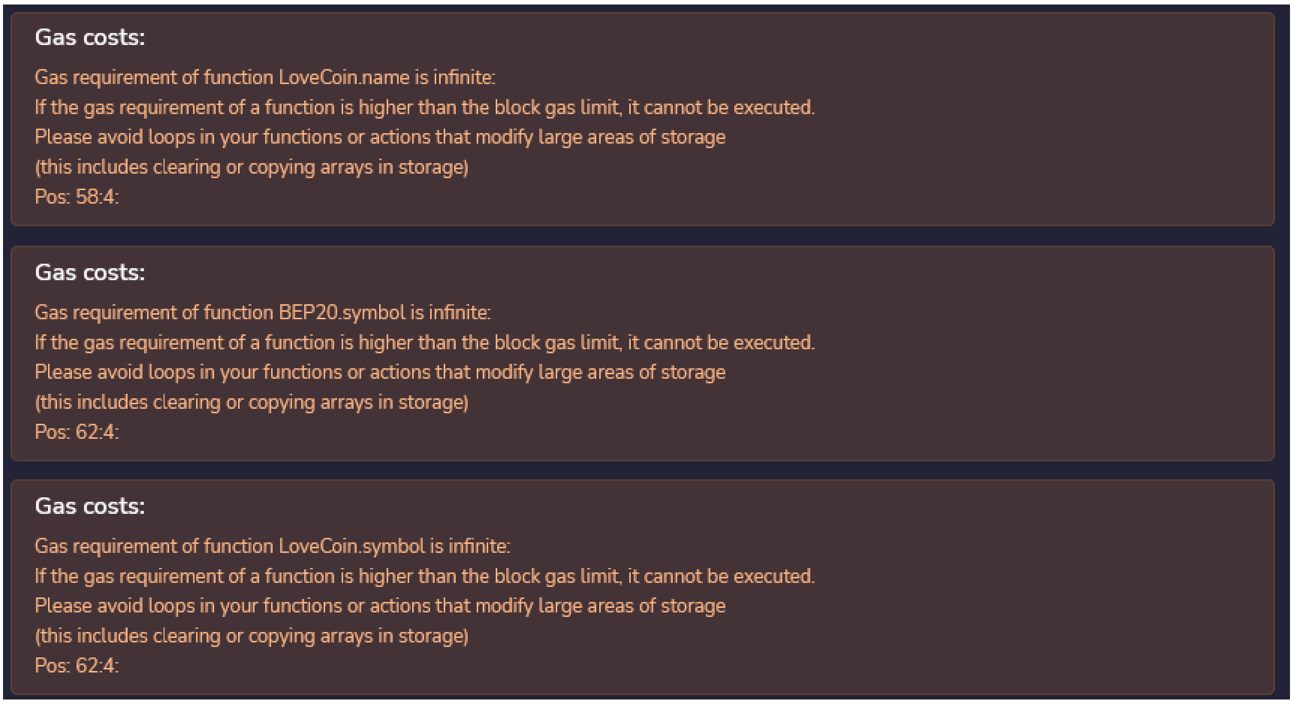
**Block timestamp:**

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 422:16:

Gas & Economy

**Gas costs:**

Gas requirement of function BEP20.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: 58:4:



**Gas costs:**

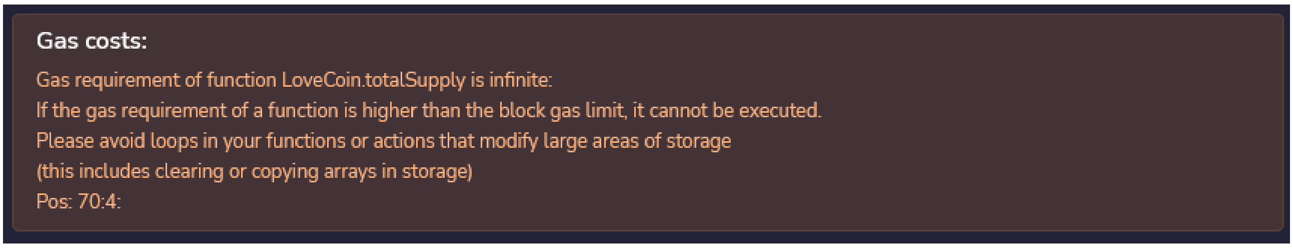
Gas requirement of function LoveCoin.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 58:4:

**Gas costs:**

Gas requirement of function BEP20.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: 62:4:

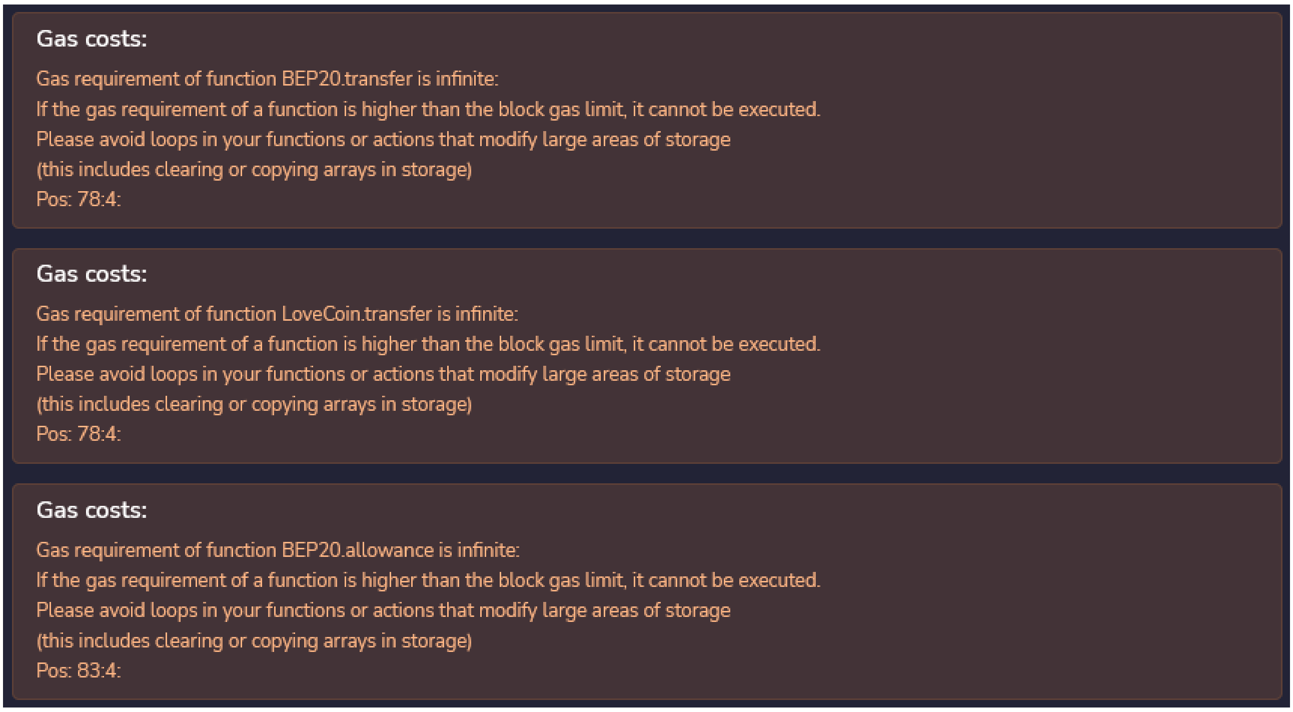
**Gas costs:**

Gas requirement of function LoveCoin.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: 62:4:



**Gas costs:**

Gas requirement of function LoveCoin.totalSupply 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: 70:4:



**Gas costs:**

Gas requirement of function BEP20.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: 78:4:

**Gas costs:**

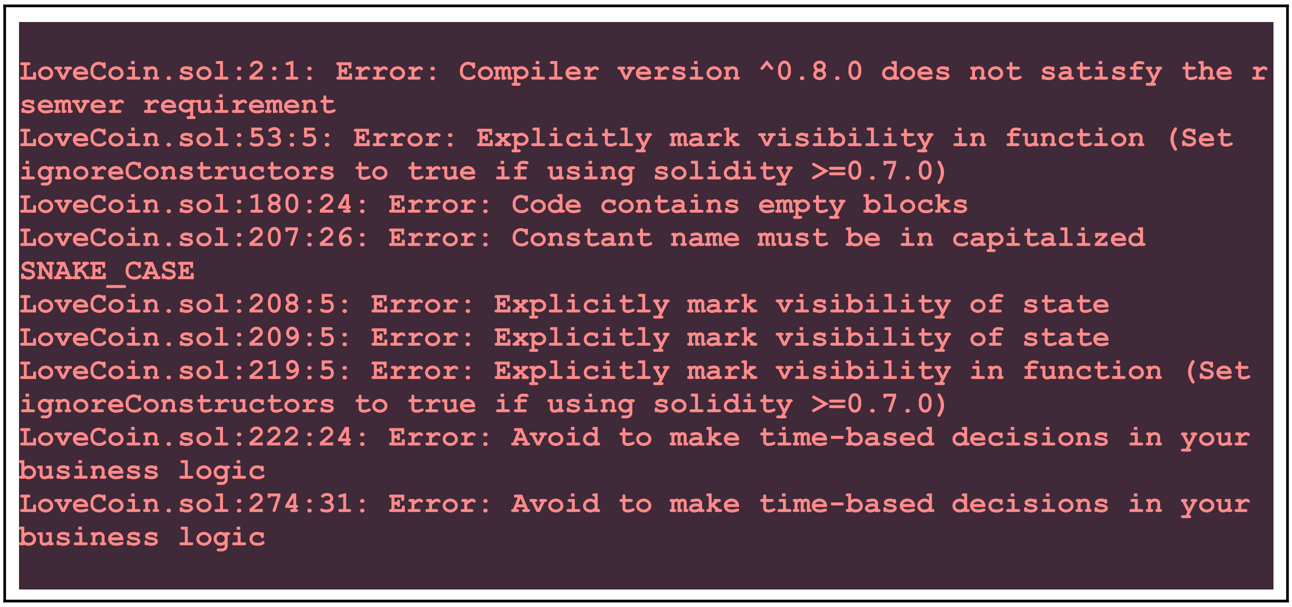
Gas requirement of function LoveCoin.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 78:4:

**Gas costs:**

Gas requirement of function BEP20.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: 83:4:

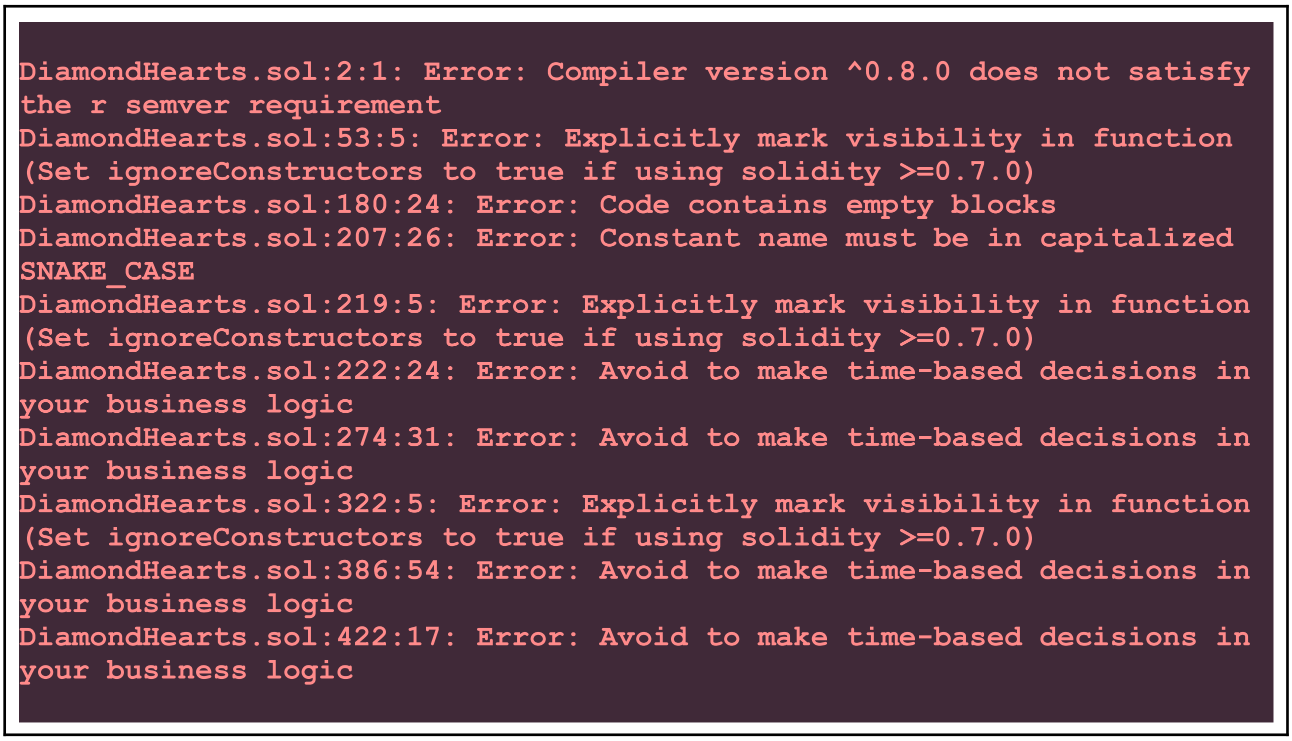
# Solhint Linter

### LoveCoin.sol



```
LoveCoin.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy the r
semver requirement
LoveCoin.sol:53:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
LoveCoin.sol:180:24: Error: Code contains empty blocks
LoveCoin.sol:207:26: Error: Constant name must be in capitalized
SNAKE_CASE
LoveCoin.sol:208:5: Error: Explicitly mark visibility of state
LoveCoin.sol:209:5: Error: Explicitly mark visibility of state
LoveCoin.sol:219:5: Error: Explicitly mark visibility in function (Set
ignoreConstructors to true if using solidity >=0.7.0)
LoveCoin.sol:222:24: Error: Avoid to make time-based decisions in your
business logic
LoveCoin.sol:274:31: Error: Avoid to make time-based decisions in your
business logic
```

### DiamondHearts.sol



```
DiamondHearts.sol:2:1: Error: Compiler version ^0.8.0 does not satisfy
the r semver requirement
DiamondHearts.sol:53:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DiamondHearts.sol:180:24: Error: Code contains empty blocks
DiamondHearts.sol:207:26: Error: Constant name must be in capitalized
SNAKE_CASE
DiamondHearts.sol:219:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DiamondHearts.sol:222:24: Error: Avoid to make time-based decisions in
your business logic
DiamondHearts.sol:274:31: Error: Avoid to make time-based decisions in
your business logic
DiamondHearts.sol:322:5: Error: Explicitly mark visibility in function
(Set ignoreConstructors to true if using solidity >=0.7.0)
DiamondHearts.sol:386:54: Error: Avoid to make time-based decisions in
your business logic
DiamondHearts.sol:422:17: Error: Avoid to make time-based decisions in
your business logic
```

**Software analysis result:**

These software reported many false positive results and some are informational issues. So, those issues can be safely ignored.

This is a private and confidential document. No part of this document should be disclosed to third party without prior written permission of EtherAuthority.

**Email: audit@EtherAuthority.io**