auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

LION

DATED : 07 June 24'

AUDIT SUMMARY

Project name – LION

Date: 07 June, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: PASSED

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	2	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0xbed9b9aab5e2492261899a49c24ebf203f69f6db#code

Token Information

Token Address:
0x2Abb5adC88c82459C72EE15aB5291c3f417118Da

Name: LION

Symbol: LION

Decimals: 18

Network: EtherScan

Token Type: ERC-20

Owner: 0xe4bE0DE4e4f10EfE06Ec8d1f02709706FBEB85aE

Deployer: 0xe4bE0DE4e4f10EfE06Ec8d1f02709706FBEB85aE

Token Supply: 50000000

Checksum: Ac6659e84744e0102ab19c1d1e78a223

Testnet:
https://testnet.bscscan.com/address/0xbed9b9aab5e2492261899a49c24ebf203f69f6db#code

TOKEN OVERVIEW

Buy Fee: 0%

Sell Fee: 0%

Transfer Fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: No

AUDIT METHODOLOGY

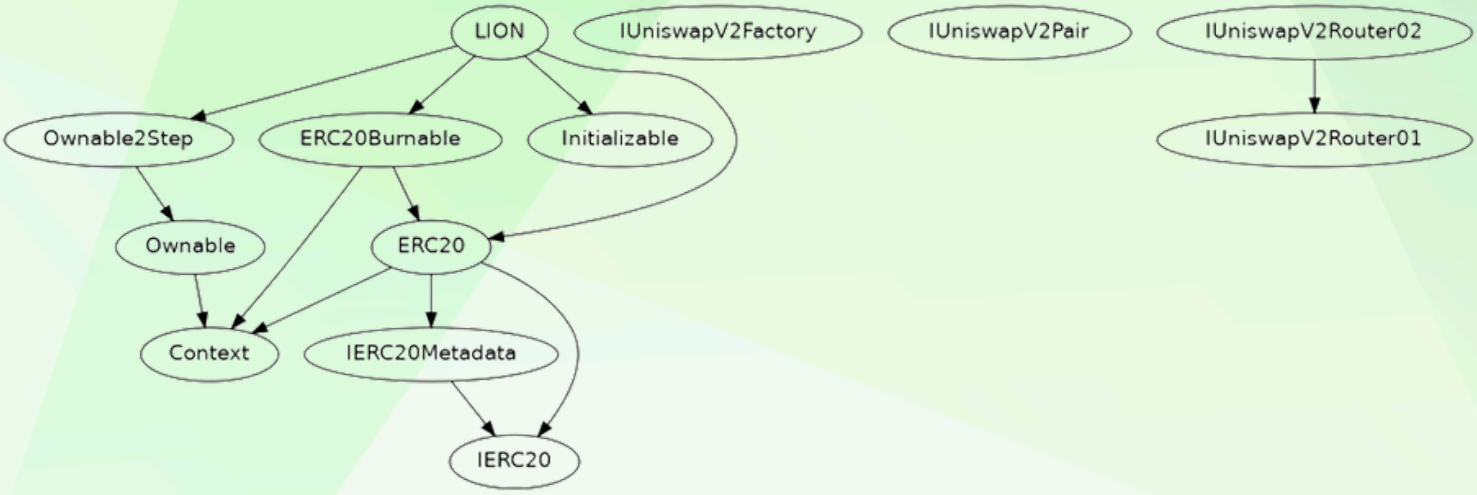
The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

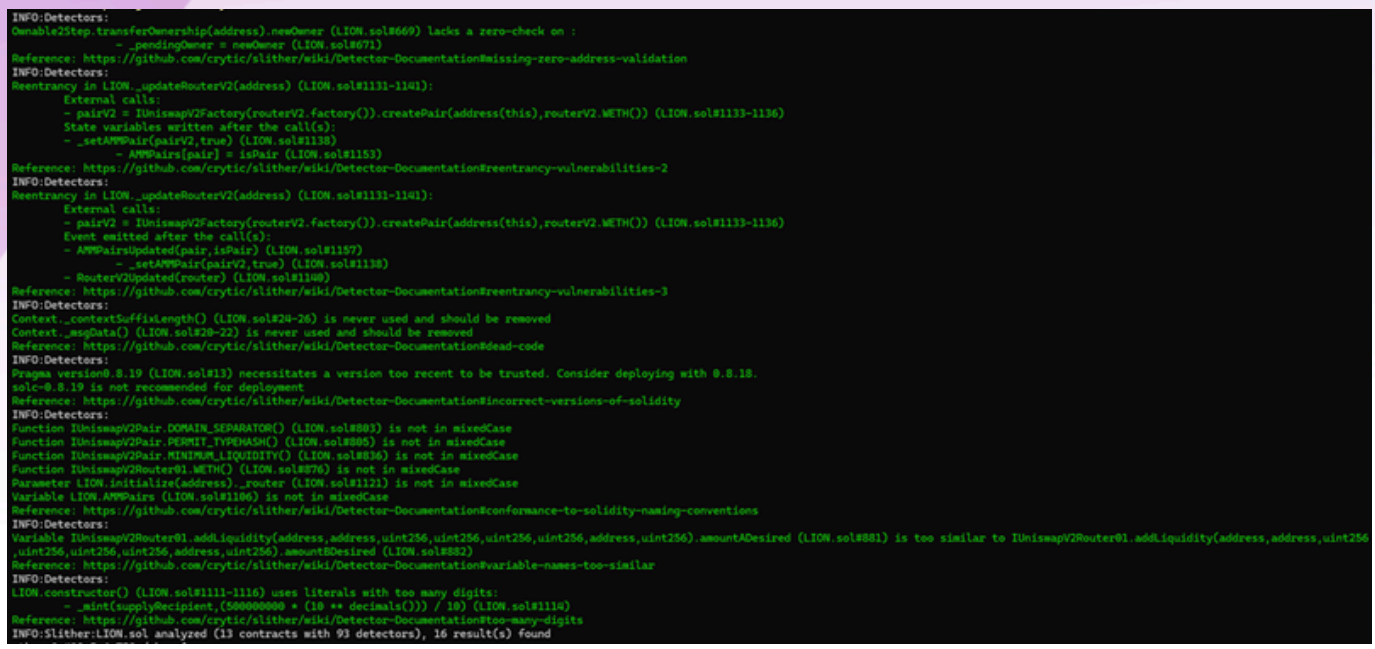
INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set the AMMPair address.

STATIC ANALYSIS



```
INFO:Detectors:
Ownable2Step.transferOwnership(address).newOwner (LION.sol#669) lacks a zero-check on :
        - _pendingOwner = newOwner (LION.sol#671)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in LION._updateRouterV2(address) (LION.sol#1131-1141):
        External calls:
        - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(address(this),routerV2.WETH()) (LION.sol#1133-1136)
        State variables written after the call(s):
        - _setAMMPair(pairV2,true) (LION.sol#1138)
                - AMMPairs[pair] = isPair (LION.sol#1153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LION._updateRouterV2(address) (LION.sol#1131-1141):
        External calls:
        - pairV2 = IUniswapV2Factory(routerV2.factory()).createPair(address(this),routerV2.WETH()) (LION.sol#1133-1136)
        Event emitted after the call(s):
        - AMMPairsUpdated(pair,isPair) (LION.sol#1157)
                - _setAMMPair(pairV2,true) (LION.sol#1138)
        - RouterV2Updated(router) (LION.sol#1140)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._contextSuffixLength() (LION.sol#24-26) is never used and should be removed
Context._msgData() (LION.sol#20-22) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.8.19 (LION.sol#13) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (LION.sol#803) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (LION.sol#805) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (LION.sol#836) is not in mixedCase
Function IUniswapV2Router01.WETH() (LION.sol#876) is not in mixedCase
Parameter LION.initialize(address)._router (LION.sol#1121) is not in mixedCase
Variable LION.AMMPairs (LION.sol#1106) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (LION.sol#881) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256
,uint256,uint256,uint256,address,uint256).amountBDesired (LION.sol#882)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
LION.constructor() (LION.sol#1111-1116) uses literals with too many digits:
        - _mint(supplyRecipient,(500000000 * (10 ** decimals())) / 10) (LION.sol#1114)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:LION.sol analyzed (13 contracts with 93 detectors), 16 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0x405a74306f1dbdd564ec141afbab6a5486e358a33027312a7f933cf3e30a45fa

2- Increase Allowance (passed):

https://testnet.bscscan.com/tx/0x3a0d0bc559735fc2f66ede43d026a95978473781b03ae20b1febe3e434bd5d08

3- Decrease Allowance (passed):

https://testnet.bscscan.com/tx/0xd3a9f6e9980cfe56026c08f56ceb2810a12ec87c08914b7aa11d427754275442

4- Burn (passed):

https://testnet.bscscan.com/tx/0x26cc8a9b25d28ac3133e9a7e96e89fac9c67b2a68335d8d43b80028f61fbb9bc

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	0
Medium-Risk	0
Low-Risk	2
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setAMMPair(address pair, bool isPair) external onlyOwner {
    require(
      pair != pairV2,
      "DefaultRouter: Cannot remove initial pair from list"
    );

    _setAMMPair(pair, isPair);
  }
function initialize(address _router) external initializer {
    _updateRouterV2(_router);
  }
```

Suggestion:

Emit an event for critical changes.

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Subject: Zero Check

Status: Open

Overview:

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function initialize(address _router) external initializer {
   _updateRouterV2(_router);
 }
```

MANUAL TESTING

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them.

```
function _msgData() internal view virtual returns (bytes calldata) {
    return msg.data;
  }
function _contextSuffixLength() internal view virtual returns (uint256) {
    return 0;
  }
interface IUniswapV2Pair
```

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace