auditace.tech

AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Trust launchpad

DATED : 08 April 24

AUDIT SUMMARY

Project name – Trust launchpad

Date: 08 April 24

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	1	0	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/0x8ada0fbc158e64a80e2037ec72d7646b4dd469f4#code

Token Information

Token Address:
0x285B4E9D54d5D419FABD904A8129Cc822C3d894e

Name: Trust launchpad

Symbol: Trust

Decimals: 9

Network: BscScan

Token Type: BEP-20

Owner: 0x51c17932cD067e3bdF01a31b54b83De39C7E1F48

Deployer: 0x51c17932cD067e3bdF01a31b54b83De39C7E1F48

Token Supply: 1,000,000,000

Checksum: Ac6659e84744e0102ab19c1d1e78a221

Testnet:
https://testnet.bscscan.com/address/0x8ada0fbc158e64a80e2037ec72d7646b4dd469f4#code

TOKEN OVERVIEW

Reflection Tax:	0-20%
Treasury Tax:	3-20%
Transfer Fee:	0-0%
Fee Privilege:	Owner
Ownership:	Owned
Minting:	No
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.

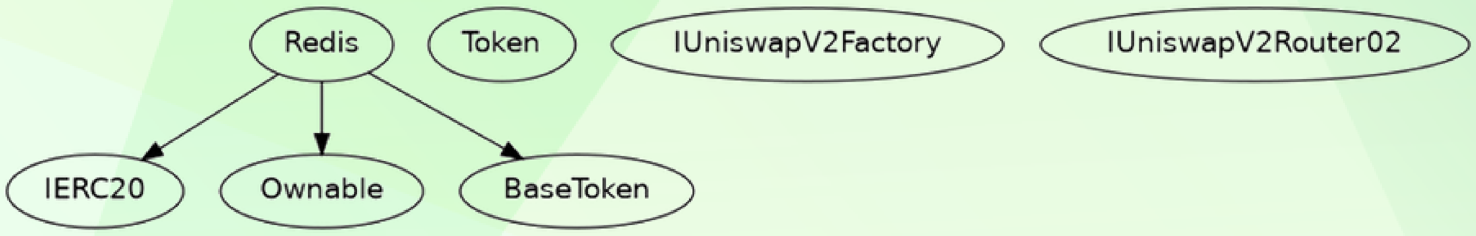
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

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

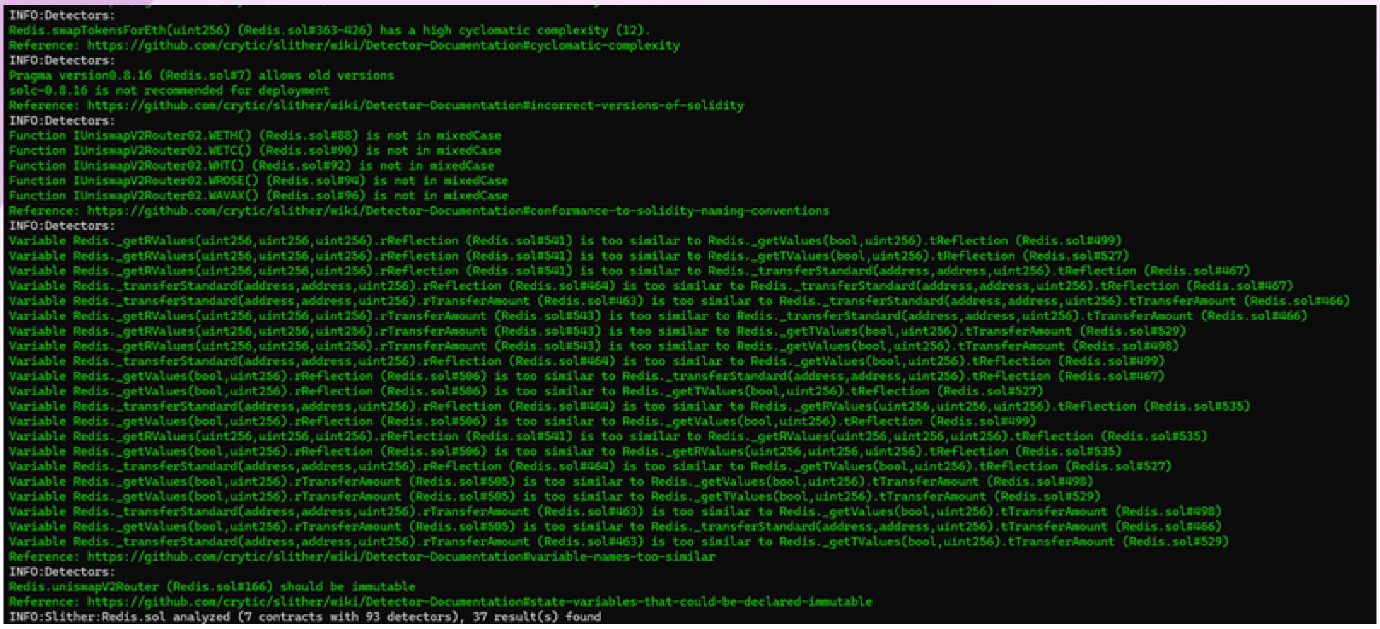
INHERITANCE TREE



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can withdraw ETH.
- The owner can set the Treasury address.
- The owner can manually swap.
- The owner can set the fee not to more than 20%.
- The owner can exclude multiple addresses from fees.

STATIC ANALYSIS



```
INFO:Detectors:
Redis.swapTokensForEth(uint256) (Redis.sol#363-426) has a high cyclomatic complexity (12).
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#cyclomatic-complexity
INFO:Detectors:
Pragma version0.8.16 (Redis.sol#7) allows old versions
solc-0.8.16 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function IUniswapV2Router02.WETH() (Redis.sol#88) is not in mixedCase
Function IUniswapV2Router02.WETC() (Redis.sol#90) is not in mixedCase
Function IUniswapV2Router02.WHT() (Redis.sol#92) is not in mixedCase
Function IUniswapV2Router02.WROSE() (Redis.sol#94) is not in mixedCase
Function IUniswapV2Router02.WAVAX() (Redis.sol#96) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable Redis._getRValues(uint256,uint256,uint256).rReflection (Redis.sol#541) is too similar to Redis._getValues(bool,uint256).tReflection (Redis.sol#499)
Variable Redis._getRValues(uint256,uint256,uint256).rReflection (Redis.sol#541) is too similar to Redis._getTValues(bool,uint256).tReflection (Redis.sol#527)
Variable Redis._getRValues(uint256,uint256,uint256).rReflection (Redis.sol#541) is too similar to Redis._transferStandard(address,address,uint256).tReflection (Redis.sol#467)
Variable Redis._transferStandard(address,address,uint256).rReflection (Redis.sol#464) is too similar to Redis._transferStandard(address,address,uint256).tReflection (Redis.sol#467)
Variable Redis._transferStandard(address,address,uint256).rTransferAmount (Redis.sol#463) is too similar to Redis._transferStandard(address,address,uint256).tTransferAmount (Redis.sol#466)
Variable Redis._getRValues(uint256,uint256,uint256).rTransferAmount (Redis.sol#543) is too similar to Redis._transferStandard(address,address,uint256).tTransferAmount (Redis.sol#466)
Variable Redis._getRValues(uint256,uint256,uint256).rTransferAmount (Redis.sol#543) is too similar to Redis._getTValues(bool,uint256).tTransferAmount (Redis.sol#529)
Variable Redis._getRValues(uint256,uint256,uint256).rTransferAmount (Redis.sol#543) is too similar to Redis._getValues(bool,uint256).tTransferAmount (Redis.sol#498)
Variable Redis._transferStandard(address,address,uint256).rReflection (Redis.sol#464) is too similar to Redis._getValues(bool,uint256).tReflection (Redis.sol#499)
Variable Redis._getValues(bool,uint256).rReflection (Redis.sol#506) is too similar to Redis._transferStandard(address,address,uint256).tReflection (Redis.sol#467)
Variable Redis._getValues(bool,uint256).rReflection (Redis.sol#506) is too similar to Redis._getTValues(bool,uint256).tReflection (Redis.sol#527)
Variable Redis._transferStandard(address,address,uint256).rReflection (Redis.sol#464) is too similar to Redis._getRValues(uint256,uint256,uint256).tReflection (Redis.sol#535)
Variable Redis._getValues(bool,uint256).rReflection (Redis.sol#506) is too similar to Redis._getValues(bool,uint256).tReflection (Redis.sol#499)
Variable Redis._getRValues(uint256,uint256,uint256).rReflection (Redis.sol#541) is too similar to Redis._getRValues(uint256,uint256,uint256).tReflection (Redis.sol#535)
Variable Redis._getValues(bool,uint256).rReflection (Redis.sol#506) is too similar to Redis._getRValues(uint256,uint256,uint256).tReflection (Redis.sol#535)
Variable Redis._transferStandard(address,address,uint256).rReflection (Redis.sol#464) is too similar to Redis._getTValues(bool,uint256).tReflection (Redis.sol#527)
Variable Redis._getValues(bool,uint256).rTransferAmount (Redis.sol#505) is too similar to Redis._getValues(bool,uint256).tTransferAmount (Redis.sol#498)
Variable Redis._getValues(bool,uint256).rTransferAmount (Redis.sol#505) is too similar to Redis._getTValues(bool,uint256).tTransferAmount (Redis.sol#529)
Variable Redis._transferStandard(address,address,uint256).rTransferAmount (Redis.sol#463) is too similar to Redis._getValues(bool,uint256).tTransferAmount (Redis.sol#498)
Variable Redis._getValues(bool,uint256).rTransferAmount (Redis.sol#505) is too similar to Redis._transferStandard(address,address,uint256).tTransferAmount (Redis.sol#466)
Variable Redis._transferStandard(address,address,uint256).rTransferAmount (Redis.sol#463) is too similar to Redis._getTValues(bool,uint256).tTransferAmount (Redis.sol#529)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Redis.uniswapV2Router (Redis.sol#166) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:Redis.sol analyzed (7 contracts with 93 detectors), 37 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/0x3d3124c7565e8cd103e8cbc790e316f93cfce26993aa6bef3a79089100cbdf33

2- Exclude Multiple Accounts From Fees (passed):

https://testnet.bscscan.com/tx/0x9d838b3f4855398cd22a51d03f58d53557604cccae4be0f07ef8c6bd478add92

3- Set Fee (passed):

https://testnet.bscscan.com/tx/0x44bf02d2315871aa7d335ea2a346b7079a782346c30f3da767a1ab1f1e27c29f

4- Set Treasury Address (passed):

https://testnet.bscscan.com/tx/0x2d92e9a1e0ca04922783b10961cfa3e6fd4b561bcd723060d3a1c6bdd3849af5

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	1
♦ Low-Risk	0
♦ Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function: setTreasuryAddress

Status: Open

Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setTreasuryAddress(address payable account) external
onlyOwner {
    require(account != address(0x0), "treasury address cannot be
zero");

    treasuryAddress = account;
    _isExcludedFromFee[account] = true;

    emit UpdatedTreasuryWallet(account);
  }
```

Suggestion:

It is recommended that the address should not be able to be set as a contract address.

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.

```
interface Token {
  function transferFrom(address, address, uint) external
returns (bool);

  function transfer(address, uint) external returns (bool);
}
```

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