# IDO AvaXLauncher Smart Contract Final Audit Report

SEP 2021

# CONTENT

## SCOPE OF AUDIT

The scope of this audit was to analyze and document the AvaXLauncher IDO 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
- ERC-20 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 BEP-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.

## Static Analysis

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

## Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities 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 the automated analysis were manually verified.

## Gas Consumption

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

## Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis, Theo.

## ISSUE CATEGORIES

Every issue in this report has been assigned to 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 be fixed before moving to a live environment.

### Medium 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 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 Issues

These are four severity issues that 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 SECURITY ISSUES PER SEVERITY

| TYPE | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| Open | 0 | 0 | 0 | 1 |
| Acknowledged | 0 | 0 | 0 | 1 |
| Closed | 0 | 0 | 0 | 1 |

# Introduction

During the period of September 02, 2021 to September 06, 2021 - Crypticocean Team performed a security audit for AvaXLauncher smart contracts.

## ISSUES FOUND – CODE REVIEW / MANUAL TESTING

No issues were found.

| | |
|---|---|
| High Severity Issues | No issues were found |
| Medium Severity Issues | No issues were found |
| Low Severity Issues | No issues were found |

### Informational Issues

A.1 Public function that could be declared external

The following public functions that are never called by the contract should be declared external to save gas:

- AvaxLancherIDO.isUserWhitelisted (AvaxIDO.sol#267-272) should be declared external
- AvaxLancherIDO.depositIDO (AvaxIDO.sol#274-286) should be declared external
- AvaxLancherIDO.userInvestment (AvaxIDO.sol#289-294) should be declared external
- AvaxLancherIDO.contractStats (AvaxIDO.sol#296-301) should be declared external
- AvaxLancherIDO.transferAnyERC20Token (AvaxIDO.sol#304-307) should be declared external

**Remediation**
Use the external attribute for functions that are never called from the contract.

**Status: Closed**

# FUNCTIONAL TESTS

| Function Name() | Technical Result | Logical Result | Overall Result |
|---|---|---|---|
| **Read Functions()** | | | |
| contractStats | Pass | Pass | Pass |
| participants | Pass | Pass | Pass |
| totalInvestment | Pass | Pass | Pass |
| totalExpected | Pass | Pass | Pass |
| wallet | Pass | Pass | Pass |
| owner | Pass | Pass | Pass |
| **Write Functions()** | | | |
| depositIDO | Pass | Pass | Pass |
| authorizeKYC | Pass | Pass | Pass |
| transferOwnership | Pass | Pass | Pass |
| pause | Pass | Pass | Pass |
| transferAnyERC20 | Pass | Pass | Pass |

# UNIT TESTS

- ✓ Should correctly initialize constructor values of USDT Dummy Token Contract (79ms)
- ✓ Should correctly initialize constructor values of AvaxLancher IDO contract (84ms)
- ✓ Should check a name of a token USDT Dummy(44ms)
- ✓ Should check a symbol of a token USDT
- ✓ Should check a decimal of a token USDT
- ✓ Should check an owner of a token
- ✓ Should check a balance of a token contract
- ✓ Should check maximum USDT allowed
- ✓ Should check USDT deposit by the user when not deposit
- ✓ Should check if the user is participant when user is not
- ✓ Should check wallet address for collection of USDT
- ✓ Should check wallet address for collection of USDT
- ✓ Should check total Investment till now in USDT
- ✓ Should address is whitelisted or not
- ✓ Should check total USDT to raise
- ✓ Should check Contract stats
- ✓ Should Not be able to whitelist by NON owner account (55ms)
- ✓ Should Not be able to pause the contract by non owner account (44ms)
- ✓ Should be able to pause the contract (45ms)
- ✓ Should check if contract is paused or not after pause
- ✓ Should Not be able to whitelist when contract is paused (50ms)
- ✓ Should address is whitelisted or not when done in pause state
- ✓ Should Not be able to unpause the contract by non owner account (42ms)
- ✓ Should be able to unpause the contract from Owner Account (44ms)
- ✓ Should check if contract is paused or not after unpaused

- ✓ Should be able to whitelist (50ms)
- ✓ Should address is whitelisted accounts[2]
- ✓ Should address is whitelisted accounts[3]
- ✓ Should address is whitelisted accounts[4]
- ✓ Should Not be able to participate in IDO if not whitelisted (48ms)
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should be able to transfer USDT (45ms)
- ✓ Should be able to transfer USDT (52ms)
- ✓ Should be able to transfer USDT (43ms)
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should check a USDT balance of accounts before transferring tokens
- ✓ Should check approval by accounts 0 to accounts 1 to spend tokens on the behalf of accounts 0
- ✓ Should Approve IDO to spend specific tokens of accounts[2]
- ✓ Should be able to participate in IDO (94ms)
- ✓ Should check if the user is a participant, when user is
- ✓ Should check Contract stats
- ✓ Should check a USDT balance of accounts after IDO participant
- ✓ Should check a USDT balance of a wallet after IDO participant
- ✓ Should Not be able to participate in IDO if once done by whitelisted (53ms)
- ✓ Should check approval by accounts 3 to IDO Contract to spend tokens on the behalf of accounts
- ✓ Should Approve IDO to spend specific tokens of accounts[3]
- ✓ Should be able to participate in IDO (95ms)
- ✓ Should check if user is participant, when user is
- ✓ Should check Contract stats

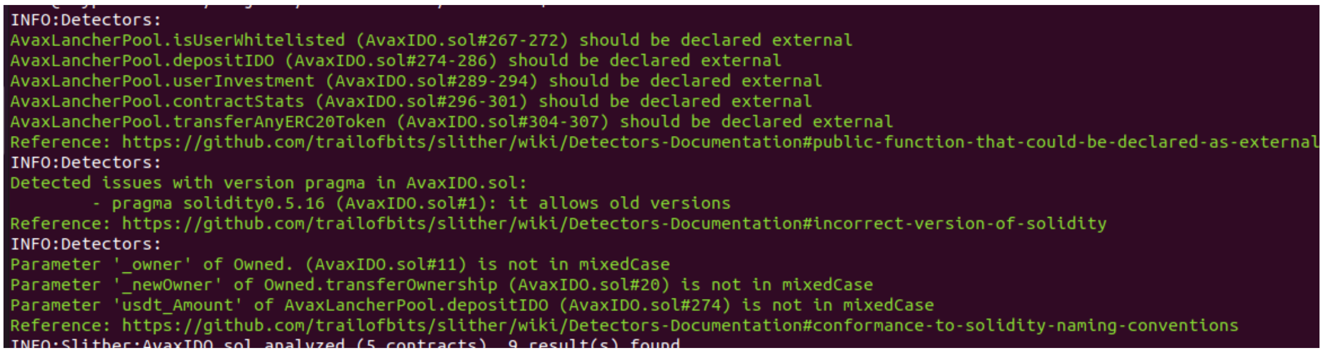
- ✓ Should check a USDT balance of accounts after IDO participant
- ✓ Should check a USDT balance of a wallet after IDO participant
- ✓ Should Not be able to participate in IDO if once done by whitelisted (40ms)
- ✓ Should check approval by accounts 3 to IDO Contract to spend tokens on the
- ✓ behalf of accounts 3
- ✓ Should Approve IDO to spend specific tokens of accounts[4]
- ✓ Should be able to participate in IDO (92ms)
- ✓ Should check if user is participant, when user is
- ✓ Should check Contract stats
- ✓ Should check a USDT balance of accounts after IDO participant
- ✓ Should check a USDT balance of a wallet after IDO participant
- ✓ Should Not be able to participate in IDO if once done by whitelisted (45ms)

**67 passing (3s)**

**0 Failed**

# AUTOMATED TESTS

## Slither:

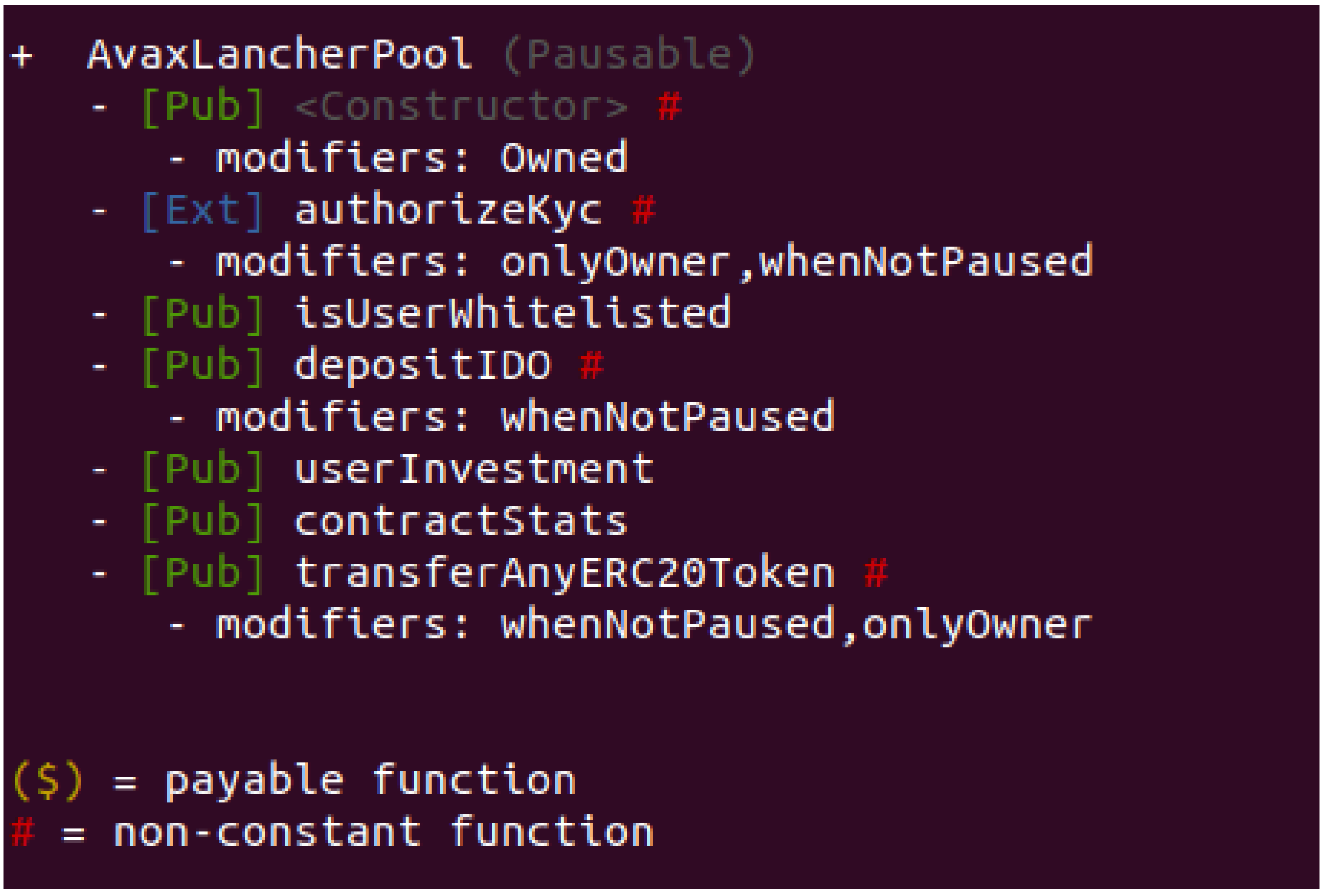


```
INFO:Detectors:
AvaxLancherPool.isUserWhitelisted (AvaxIDO.sol#267-272) should be declared external
AvaxLancherPool.depositIDO (AvaxIDO.sol#274-286) should be declared external
AvaxLancherPool.userInvestment (AvaxIDO.sol#289-294) should be declared external
AvaxLancherPool.contractStats (AvaxIDO.sol#296-301) should be declared external
AvaxLancherPool.transferAnyERC20Token (AvaxIDO.sol#304-307) should be declared external
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#public-function-that-could-be-declared-as-external
INFO:Detectors:
Detected issues with version pragma in AvaxIDO.sol:
        - pragma solidity0.5.16 (AvaxIDO.sol#1): it allows old versions
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#incorrect-version-of-solidity
INFO:Detectors:
Parameter '_owner' of Owned. (AvaxIDO.sol#11) is not in mixedCase
Parameter '_newOwner' of Owned.transferOwnership (AvaxIDO.sol#20) is not in mixedCase
Parameter 'usdt_Amount' of AvaxLancherPool.depositIDO (AvaxIDO.sol#274) is not in mixedCase
Reference: https://github.com/trailofbits/slither/wiki/Detectors-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:AvaxIDO.sol analyzed (5 contracts), 9 result(s) found
```

### Result:

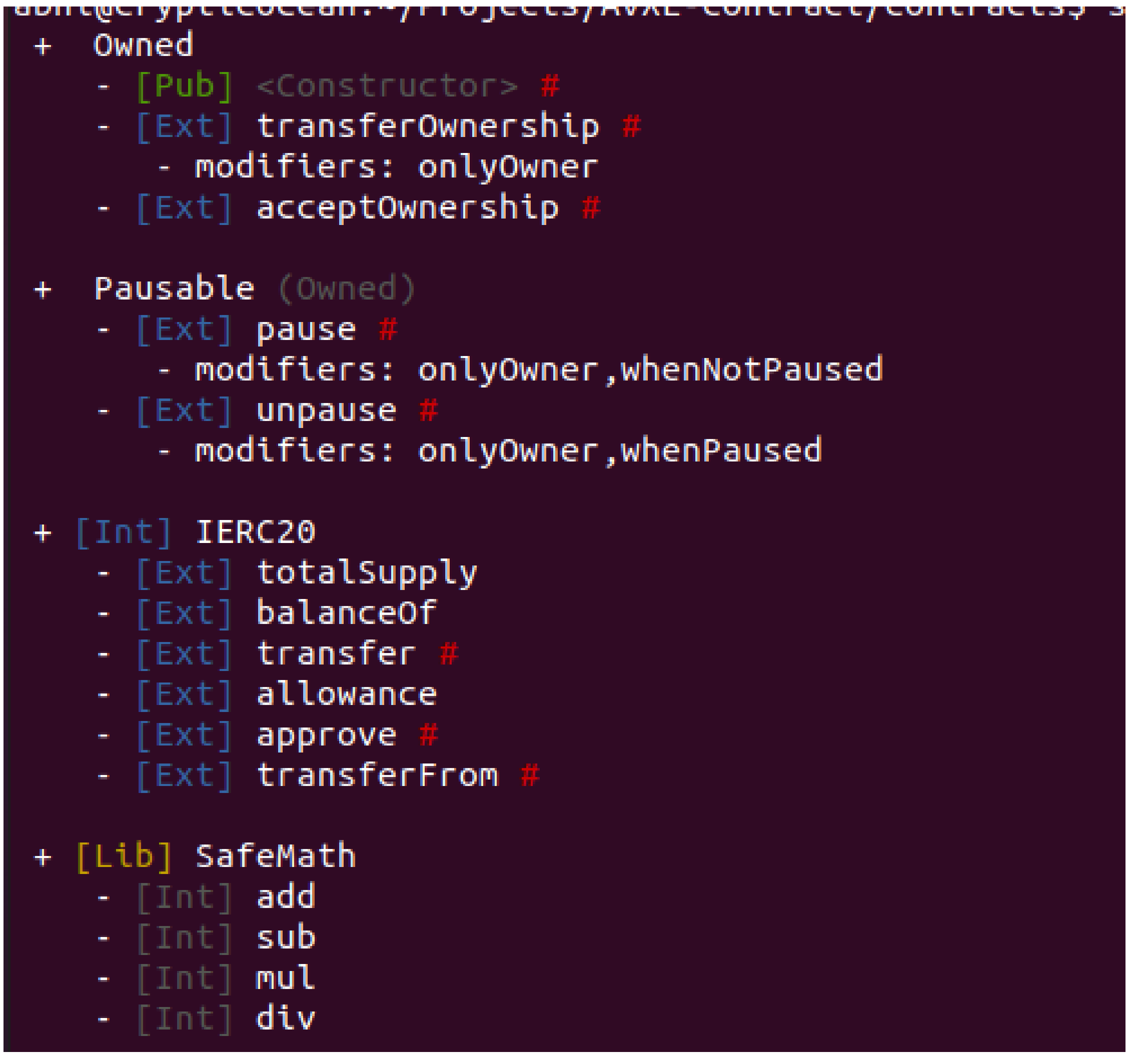
No major issues were found. Some false positive errors were reported by the tool. All the other issues have been categorized above according to their level of severity.

## Surya:



```
+  AvaxLancherPool (Pausable)
   - [Pub] <Constructor> #
      - modifiers: Owned
   - [Ext] authorizeKyc #
      - modifiers: onlyOwner,whenNotPaused
   - [Pub] isUserWhitelisted
   - [Pub] depositIDO #
      - modifiers: whenNotPaused
   - [Pub] userInvestment
   - [Pub] contractStats
   - [Pub] transferAnyERC20Token #
      - modifiers: whenNotPaused,onlyOwner


($) = payable function
# = non-constant function
```

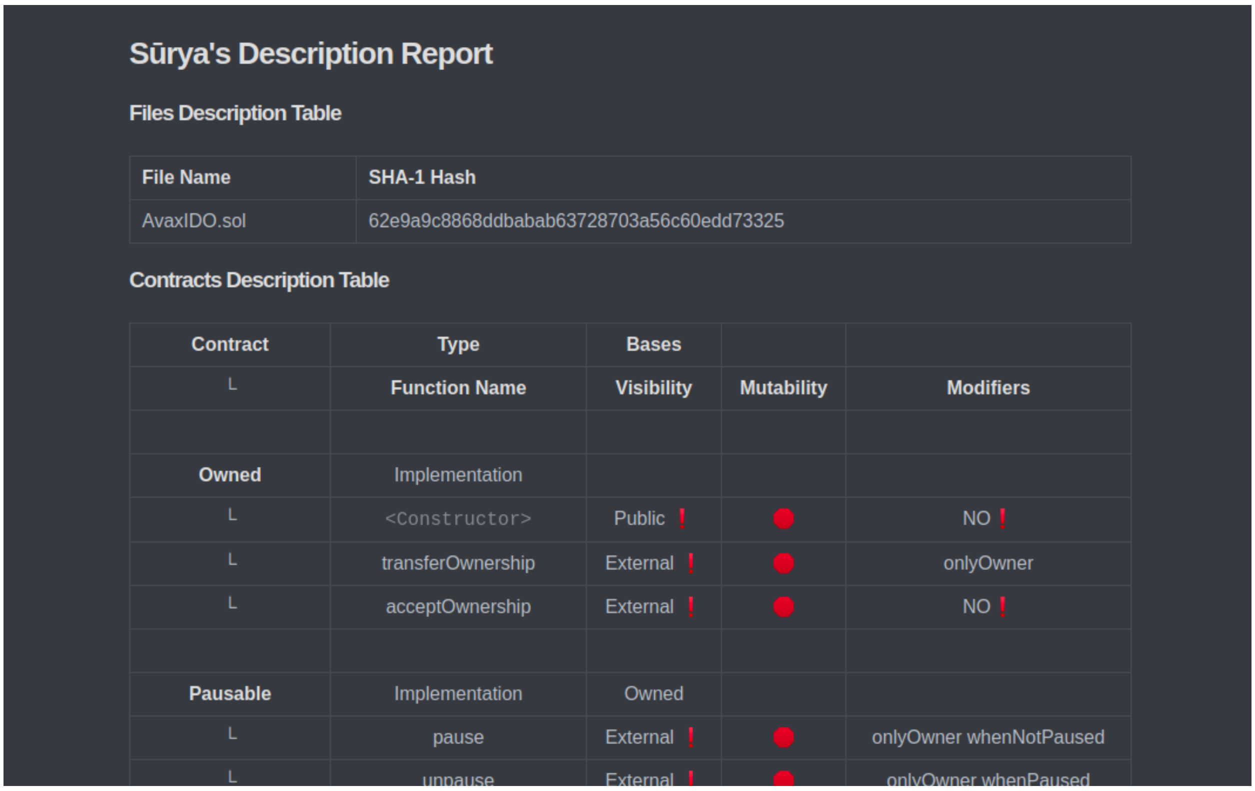


```
+  Owned
    - [Pub] <Constructor> #
    - [Ext] transferOwnership #
       - modifiers: onlyOwner
    - [Ext] acceptOwnership #

+  Pausable (Owned)
    - [Ext] pause #
       - modifiers: onlyOwner,whenNotPaused
    - [Ext] unpause #
       - modifiers: onlyOwner,whenPaused

+ [Int] IERC20
    - [Ext] totalSupply
    - [Ext] balanceOf
    - [Ext] transfer #
    - [Ext] allowance
    - [Ext] approve #
    - [Ext] transferFrom #

+ [Lib] SafeMath
    - [Int] add
    - [Int] sub
    - [Int] mul
    - [Int] div
```



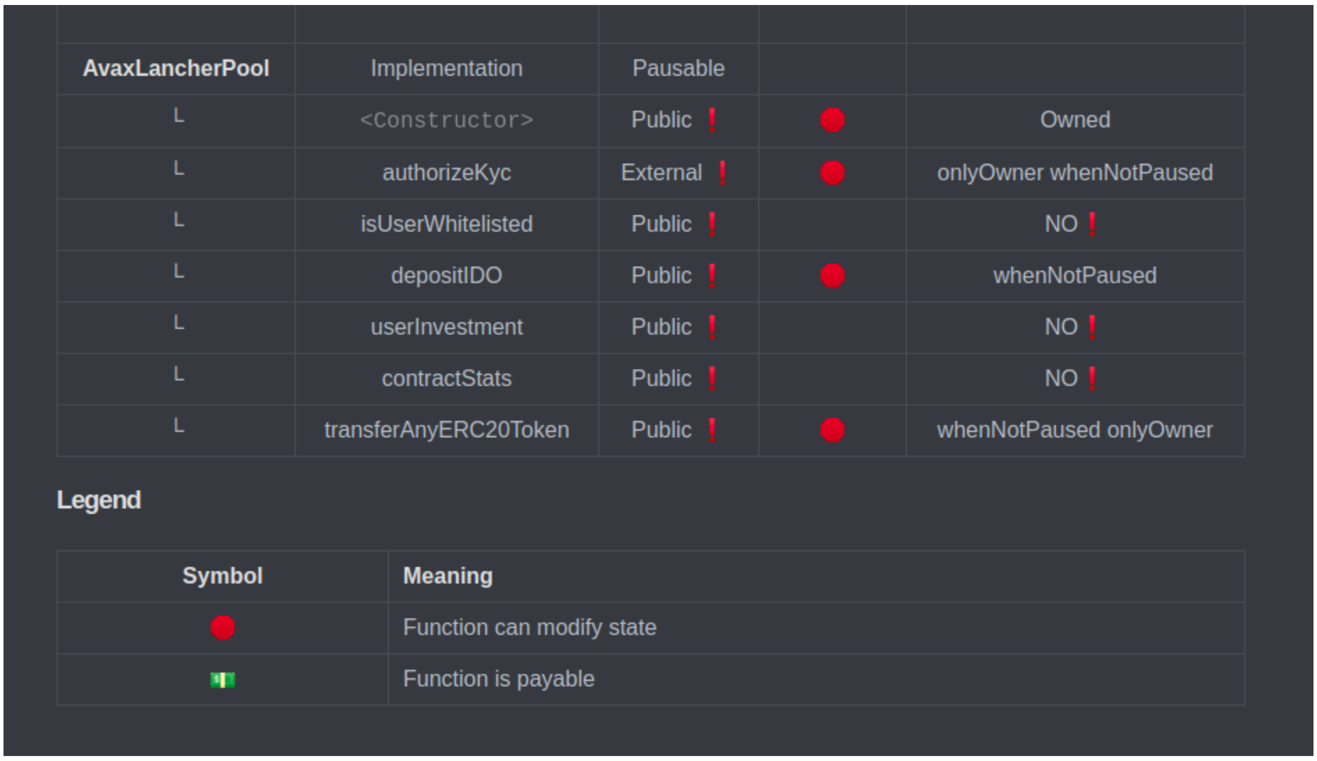
## Sūrya's Description Report

### Files Description Table

| File Name | SHA-1 Hash |
|---|---|
| AvaxIDO.sol | 62e9a9c8868ddbabab63728703a56c60edd73325 |

### Contracts Description Table

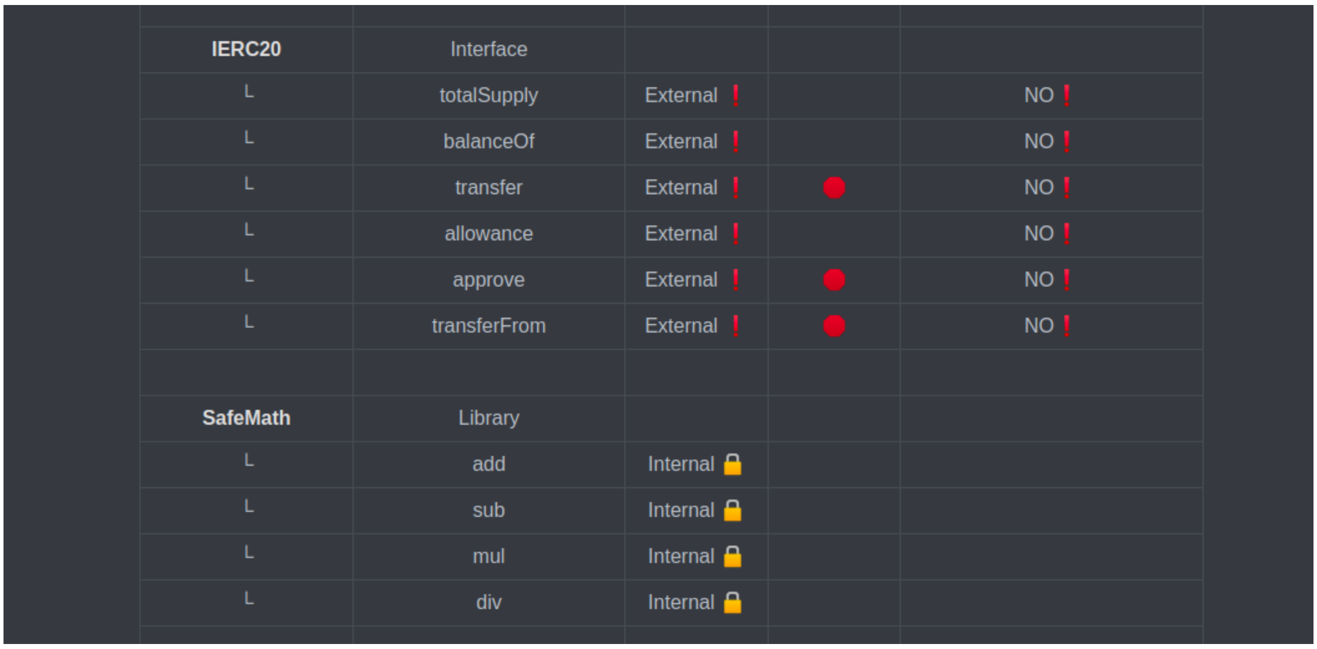
| Contract | Type | Bases | | |
|---|---|---|---|---|
| └ | Function Name | Visibility | Mutability | Modifiers |
| | | | | |
| **Owned** | Implementation | | | |
| └ | <Constructor> | Public ❗️ | 🛑 | NO❗️ |
| └ | transferOwnership | External ❗️ | 🛑 | onlyOwner |
| └ | acceptOwnership | External ❗️ | 🛑 | NO❗️ |
| | | | | |
| **Pausable** | Implementation | Owned | | |
| └ | pause | External ❗️ | 🛑 | onlyOwner whenNotPaused |
| └ | unpause | External ❗️ | 🛑 | onlyOwner whenPaused |



| | | | | |
|---|---|---|---|---|
| **AvaxLancherPool** | Implementation | Pausable | | |
| └ | `<Constructor>` | Public ❗ | 🔴 | Owned |
| └ | authorizeKyc | External ❗ | 🔴 | onlyOwner whenNotPaused |
| └ | isUserWhitelisted | Public ❗ | | NO ❗ |
| └ | depositIDO | Public ❗ | 🔴 | whenNotPaused |
| └ | userInvestment | Public ❗ | | NO ❗ |
| └ | contractStats | Public ❗ | | NO ❗ |
| └ | transferAnyERC20Token | Public ❗ | 🔴 | whenNotPaused onlyOwner |

**Legend**

| Symbol | Meaning |
|---|---|
| 🔴 | Function can modify state |
| 💵 | Function is payable |



| | | | | |
|---|---|---|---|---|
| **IERC20** | Interface | | | |
| └ | totalSupply | External ❗ | | NO ❗ |
| └ | balanceOf | External ❗ | | NO ❗ |
| └ | transfer | External ❗ | 🔴 | NO ❗ |
| └ | allowance | External ❗ | | NO ❗ |
| └ | approve | External ❗ | 🔴 | NO ❗ |
| └ | transferFrom | External ❗ | 🔴 | NO ❗ |
| | | | | |
| **SafeMath** | Library | | | |
| └ | add | Internal 🔒 | | |
| └ | sub | Internal 🔒 | | |
| └ | mul | Internal 🔒 | | |
| └ | div | Internal 🔒 | | |

## CLOSING SUMMARY

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

No instances of Integer Overflow and Underflow vulnerabilities or Back-Door Entry were found in the contract, but relying on other contracts might cause Reentrancy Vulnerability.

Some low severity issues were detected; it is recommended to fix them.

## DISCLAIMER

Cryptiocean audit is not a security warranty, investment advice, or an endorsement of the AvaXLauncher 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 AvaXLauncher Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.