auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Dollox Network

DATED : 14 April 23'

AUDIT SUMMARY

Project name – Dollox Network

Date: 14 April, 2023

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

Audit Status: Not Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	1	0	0	0	0
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 done on BSC Test network, each test has its transaction has attached to it.

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

Testnet Link:
https://testnet.bscscan.com/token/0x86a1c324D842379d4D577096768eaBff6EfC7D74

Token Information

Token Name : Dollox Network

Token Symbol: DLX

Decimals: 18

Token Supply: 1,000,000,000

Token Address:
0x82435103865fc9eEb645a6bae1F840614E0C9768

Checksum:
526329adf36e989503ef2c89d84aff5add6b9644

Owner:
0x75f58204BDa4a0CBEcBB3c2A60d0Ce26794fdA00
(at time of audit)

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: None

Ownership: Renounced

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges: Enabling trades - initializing sale and airdrops

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

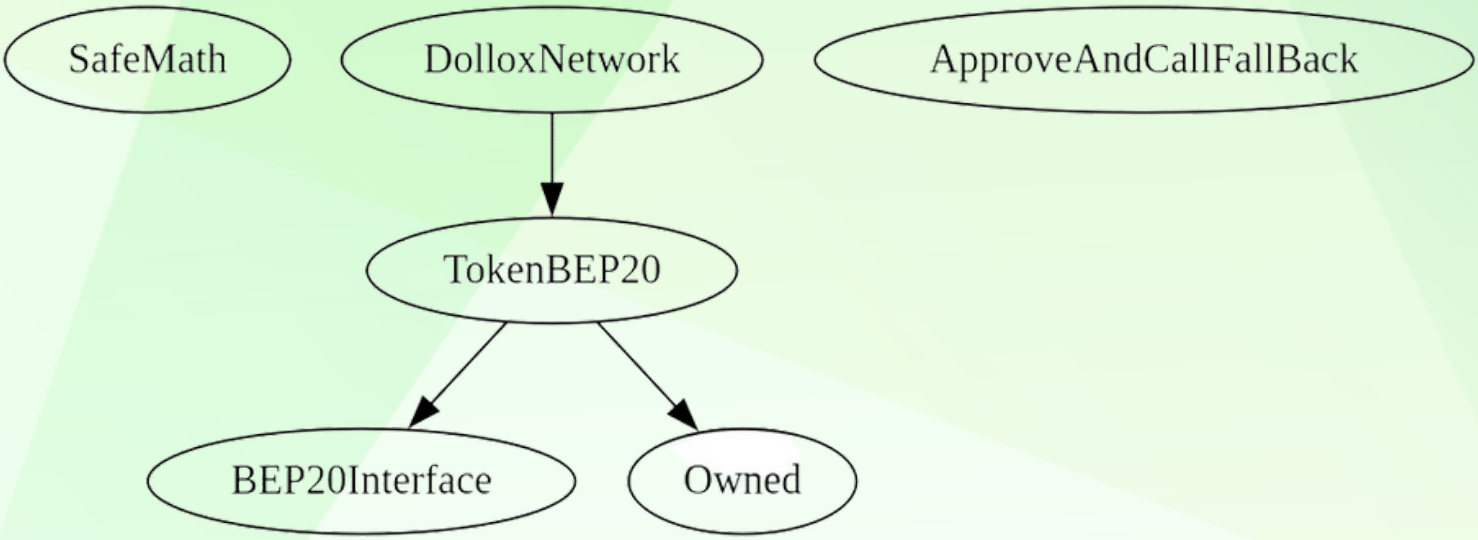
CLASSIFICATION OF RISK

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	1
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is not able to set buy/sell/transfer fees
- Owner is not able to set a max buy/transfer/wallet amount
- Owner is able to blacklist an arbitrary wallet
- Owner is able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
BEP20Interface	Implementation			
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
ApproveAndCallFallBack	Implementation			
└	receiveApproval	Public ❗	🛑	NO ❗
Owned	Implementation			
└	<Constructor>	Public ❗	🛑	NO ❗
└	transferOwnership	Public ❗	🛑	onlyOwner
└	acceptOwnership	Public ❗	🛑	NO ❗
TokenBEP20	Implementation	BEP20Interface, Owned		
└	<Constructor>	Public ❗	🛑	NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approveAndCall	Public ❗	🛑	NO ❗
└	<Fallback>	External ❗	💵	NO ❗
DolloxNetwork	Implementation	TokenBEP20		
└	<Constructor>	Public ❗	🛑	NO ❗
└	getAirdrop	Public ❗	💵	NO ❗
└	tokenSale	Public ❗	💵	NO ❗
└	viewAirdrop	Public ❗		NO ❗
└	viewSale	Public ❗		NO ❗
└	startAirdrop	Public ❗	🛑	onlyOwner

CONTRACT ASSESMENT

└	startSale	Public ❗	🛑	onlyOwner
└	clearETH	Public ❗	🛑	onlyOwner
└	<Fallback>	External ❗	💵	NO ❗
└	burn	Public ❗	🛑	onlyOwner
└	exitToken	Public ❗	🛑	onlyOwner
└	enableTrading	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Context._msgData() (bc/Context.sol#20-23) is never used and should be removed
ERC20._burn(address,uint256) (bc/ERC20.sol#289-300) is never used and should be removed
ERC20._tokengeneration(address,uint256) (bc/ERC20.sol#265-276) is never used and should be removed
SafeMath.div(uint256,uint256) (bc/SafeMath.sol#94-96) is never used and should be removed
SafeMath.div(uint256,uint256,string) (bc/SafeMath.sol#110-120) is never used and should be removed
SafeMath.mod(uint256,uint256) (bc/SafeMath.sol#134-136) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (bc/SafeMath.sol#150-157) is never used and should be removed
SafeMath.mul(uint256,uint256) (bc/SafeMath.sol#68-80) is never used and should be removed
SafeMath.sub(uint256,uint256) (bc/SafeMath.sol#33-35) is never used and should be removed
SafeMathInt.abs(int256) (bc/SafeMath.sol#212-215) is never used and should be removed
SafeMathInt.add(int256,int256) (bc/SafeMath.sol#203-207) is never used and should be removed
SafeMathInt.div(int256,int256) (bc/SafeMath.sol#183-189) is never used and should be removed
SafeMathInt.mul(int256,int256) (bc/SafeMath.sol#171-178) is never used and should be removed
SafeMathInt.sub(int256,int256) (bc/SafeMath.sol#194-198) is never used and should be removed
SafeMathInt.toUint256Safe(int256) (bc/SafeMath.sol#217-220) is never used and should be removed
SafeMathUint.toInt256Safe(uint256) (bc/SafeMath.sol#228-232) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.6 (bc/Context.sol#3) allows old versions
Pragma version^0.8.6 (bc/ERC20.sol#3) allows old versions
Pragma version^0.8.6 (bc/IERC20.sol#3) allows old versions
Pragma version^0.8.6 (bc/SafeMath.sol#3) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "this (bc/Context.sol#21)" inContext (bc/Context.sol#15-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

SafeMathInt.MAX_INT256 (bc/SafeMath.sol#166) is never used in SafeMathInt (bc/SafeMath.sol#164-221)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

ERC20._name (bc/ERC20.sol#40) should be immutable
ERC20._symbol (bc/ERC20.sol#41) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

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- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x9c38b72961a2b73bf4c94f40406c1aca67eae679dfd7a2879b0b4e0450e81c4b

2- Buying (0% tax) (passed):
https://testnet.bscscan.com/tx/0xceec4124d25ee7b491b7148cd374024bb2413e4990b030172097898ae6685420

3- Selling (0% tax) (passed):
https://testnet.bscscan.com/tx/0x2e070ec118eb69f1f8db30109de8471cff6659e457ca308c176a4646228ddb4c

4- Transferring (0% tax) (passed):
https://testnet.bscscan.com/tx/0xab6987a455d41a1303ce93064a2488e5897715cdef2d17b100ced0f60070e2cc

8- Burning (passed):
https://testnet.bscscan.com/tx/0x4d012f0b838e7d832e7738201ad6c23986a3f4cd3dd3975bd5d40f13dea47f7f

9- Token sale (passed):
https://testnet.bscscan.com/tx/0xe3ceabfe6727636345441 3d46a0154253368039 1d6351f401cef6db3d6d8c4ca

10- Airdrop (passed):
https://testnet.bscscan.com/tx/0xea0143e63f286b0e99b2b3f130640a1026a4d88ed664ec9e97386d49bd33db15

MANUAL TESTING

Logical – Invalid airdrop & sale calculations

Severity: Critical

Function: tokenSale - getAirdrop

Lines: 233, 209

Status: Not Resolved

Overview:

At tokenSale and getAirdrop functions, a portion of the airdrop/sale amount (30%) is sent to refferer, however, this amount of tokens are not deducted from total airdrop/sale amounts which could lead to invalid calculations.

On the other hand sTot (total tokens sold) and aTot (total tokens airdropped) are increasing only by 1 after each sale or airdrop. This means we will not reach the airdrop or sale cap (e.g. sale cap would be reached after 590,000,000 times contribution)

```
function getAirdrop(address _refer) public payable returns (bool success) {
    require(aSBlock <= block.number && block.number <= aEBlock);
    require(aTot < aCap || aCap == 0);
    require(_hasClaimed[msg.sender] != true, "You have already claimed!");
    require(
        msg.value >= 0.001 ether,
        "BEP20: insufficient BNB for transaction"
    );

    if (msg.value > 0.001 ether) {
        uint256 refundAmount = msg.value - 0.001 ether;
        msg.sender.transfer(refundAmount);
    }

    aTot++;
    if (
        msg.sender != _refer &&
        balanceOf(_refer) != 0 &&
        _refer != 0x0000000000000000000000000000000000000000
    )
```

MANUAL TESTING

```
 {
balances[address(this)] = balances[address(this)].sub(
(aAmt * 3) / 10
);
balances[_refer] = balances[_refer].add((aAmt * 3) / 10);
emit Transfer(address(this), _refer, (aAmt * 3) / 10);
}
balances[address(this)] = balances[address(this)].sub(aAmt);
_hasClaimed[msg.sender] = true;
balances[msg.sender] = balances[msg.sender].add(aAmt);
emit Transfer(address(this), msg.sender, aAmt);
return true;
}
```

Recommendation:

- Ensure that refferer amount is deducted from total tokens
- Ensure that total tokens sold is calculated correctly

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