AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

D TYRANT

DATED : 13 July 23'

FUNCTIONAL TESTING

Centralization – Enabling Trades

Severity: High

function: openTrade

Status: Not Resolved

Overview:

Owner of the contract must enable trades manually for investors, otherwise no one would be able to buy/sell/transfer their tokens.

```
function openTrade() external onlyOwner {
    isOpen = true;
}
```

Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a certified pinsksale safu developer to guearanee enabling of trades.

FUNCTIONAL TESTING

Centralization – Maximum transfer/buy/sell

Severity: High

function: _transferTokens

Status: Not Resolved

Overview:

only owner is able to bypass this limitation. This may cause problems if e.g. presale address has to send more tokens than maximum tx

```
function _transferTokens(address from, address to, uint256 amount) internal virtual {
    if (from != owner() && to != owner()) {
        require(amount <= _maxTxAmount, "Exceeds Max Tx Amount");
    }
    _transfer(from, to, amount);
}
```

Suggestion

Its suggested to allow whitelisted wallets to bypass this limition, this ensures that users wont have any problem claiming their tokens at time of presale.

AUDIT SUMMARY

Project name – D TYRANT

Date: 13 July, 2023

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

Audit Status: **Passed with High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	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 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/token/0x680ABEb02778dD58CEf158EB929Ab37E62fec502

Token Information

Token Name : DRAGON TYRANT

Token Symbol: D TYRANT

Decimals: 18

Token Supply: 66,666,666

Token Address:
0xc0E035e38CCdC305325aD90c0B124e364391816f

Checksum:
fa17c9f5736affaf74cb2b9343cf49e4888f335b

Owner:
0xDCCc27e1D9355651Bff0b9B1656D0C21cADd30E4
(at time of writing the audit)

Deployer:
0xDCCc27e1D9355651Bff0b9B1656D0C21cADd30E4

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: No Fees

Ownership: owned

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - Initial distribution of the tokens

- enabling trades

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

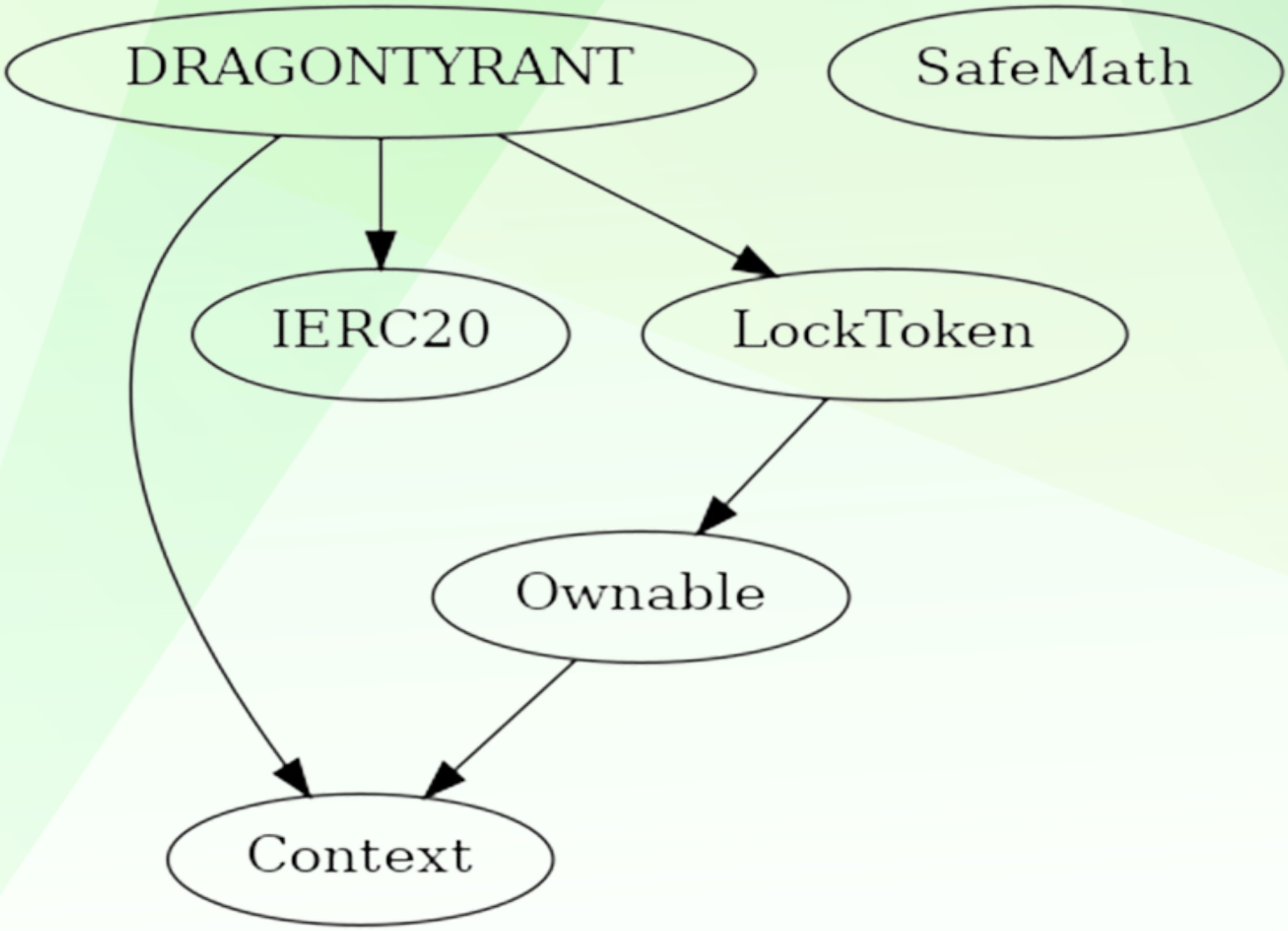
CLASSIFICATION OF RISK

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	2
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- Owner is able to set fees (static 0% fees)
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to limit buy/sell/transfer/wallet amounts
- Owner is not able to mint new tokens
- **Owner must enable trades manually**
- **there is a 4% max buy/sell/transfer**

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🛑	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🛑	NO ❗
└	transferFrom	External ❗	🛑	NO ❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
LockToken	Implementation	Ownable		
└	<Constructor>	Public ❗	🛑	NO ❗
└	openTrade	External ❗	🛑	onlyOwner
└	includeToWhiteList	External ❗	🛑	onlyOwner
DRAGONTYRANT	Implementation	Context, IERC20, LockToken		
└	<Constructor>	Public ❗	🛑	NO ❗

CONTRACT ASSESMENT

└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	_transferTokens	Internal 🔒	🛑	
└	_transfer	Internal 🔒	🛑	open
└	_mint	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
DRAGONTYRANT.allowance(address,address).owner (contracts/Token.sol#209) shadows:
        - Ownable.owner() (contracts/Token.sol#31-33) (function)
DRAGONTYRANT._approve(address,address,uint256).owner (contracts/Token.sol#264) shadows:
        - Ownable.owner() (contracts/Token.sol#31-33) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Context._msgData() (contracts/Token.sol#13-16) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#111-114) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#126-129) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#116-119) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#131-134) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#104-109) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#99-102) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#65-69) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#83-86) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#88-91) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#76-81) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#71-74) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#6) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter LockToken.includeToWhiteList(address[])._users (contracts/Token.sol#154) is not in mixedCase
Variable DRAGONTYRANT._maxTxAmount (contracts/Token.sol#172) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Redundant expression "this (contracts/Token.sol#14)" inContext (contracts/Token.sol#8-17)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

DRAGONTYRANT._decimals (contracts/Token.sol#170) should be immutable
DRAGONTYRANT._maxTxAmount (contracts/Token.sol#172) should be immutable
DRAGONTYRANT._name (contracts/Token.sol#168) should be immutable
DRAGONTYRANT._symbol (contracts/Token.sol#169) should be immutable
DRAGONTYRANT.marketingAddress (contracts/Token.sol#171) 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

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x4f1f4d26c3ff0bbb915c22af5c993ed5d4d7d9e65875c09b17f3f7360f7c569b

2- Buying (0% tax) (passed):
https://testnet.bscscan.com/tx/0x4f1f4d26c3ff0bbb915c22af5c993ed5d4d7d9e65875c09b17f3f7360f7c569b

3- Selling (0% tax) (passed):
https://testnet.bscscan.com/tx/0x62d3e4034b80c8458521ab40b05740531546880d8ade20f21ea41dcb3907d96e

4- Transferring (0% tax) (passed):
https://testnet.bscscan.com/tx/0x30f3104688ab915d59df8368834adc6473c5513a1ef5be17074e89cb250cee9b

FUNCTIONAL TESTING

Centralization – Enabling Trades

Severity: High

function: openTrade

Status: Not Resolved

Overview:

Owner of the contract must enable trades manually for investors, otherwise no one would be able to buy/sell/transfer their tokens.

```
function openTrade() external onlyOwner {
    isOpen = true;
}
```

Suggestion

Its suggested to either enable trades prior to presale, or transfer ownership of the contract to a certified pinsksale safu developer to guearanee enabling of trades.

FUNCTIONAL TESTING

Centralization – Maximum transfer/buy/sell

Severity: High

function: _transferTokens

Status: Not Resolved

Overview:

only owner is able to bypass this limitation. This may cause problems if e.g. presale address has to send more tokens than maximum tx

```
function _transferTokens(address from, address to, uint256 amount) internal virtual {
    if (from != owner() && to != owner()) {
        require(amount <= _maxTxAmount, "Exceeds Max Tx Amount");
    }
    _transfer(from, to, amount);
}
```

Suggestion

Its suggested to allow whitelisted wallets to bypass this limition, this ensures that users wont have any problem claiming their tokens at time of presale.

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