auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

ElonFloki

DATED : 30 May 23'

CRITICAL RISKS FOUND

Logical – Invalid decimal

Severity: Critical

Status: Not Resolved

Overview:

ERC20 contract which is inherited by ElonFoki token and dividend tracker, is returning “18” as decimal number. While other contracts are using “9”.

Suggestion

ERC20 must return “9” as decimal number:

```
function decimals() public view virtual override returns (uint8) {
    return 9;
}
```

HIGH RISKS FOUND

Centralization – Trades must be enabled

Severity: High

function: enableTrading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

AUDIT SUMMARY

Project name – ElonFloki

Date: 30 May, 2023

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

Audit Status: **Passed with Critical Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	1	1	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/0x5474D6D553D719298CE05972249bB5C528e665cA

Token Information

Token Name : Elon Floki CEO

Token Symbol: ElonFloki

Decimals: 9

Token Supply: 1,000,000,000

Token Address:
0x79f15BCfB0539faF42A064B841dBf9c59995350

Checksum:
ff70bb721c780c74a688fdbdfce51395e29c8d9d

Owner:
0x655776fC999B3Ece67d6124952FEdB598481A35d
(at time of writing the audit)

Deployer:
0xB2C0e01CF72D93695D60eb7DacCE85d2d40f35BF

TOKEN OVERVIEW

Fees:

Buy Fees: 0-10%

Sell Fees: 0-10%

Transfer Fees: 0-10%

Fees Privilege: Owner

Ownership: 0x655776fC999B3Ece67d6124952FEdB598481A35d

Minting: No mint function

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: including in fees

- excluding from fees
- initial distribution of the tokens
- modifying fees
- 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

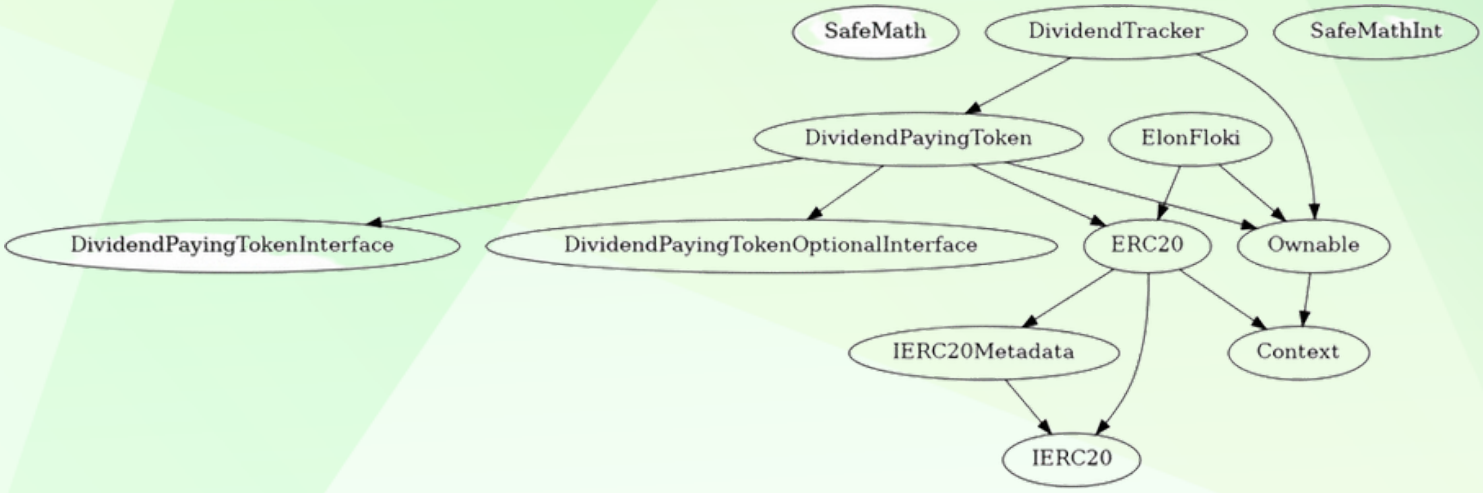
CLASSIFICATION OF RISK

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

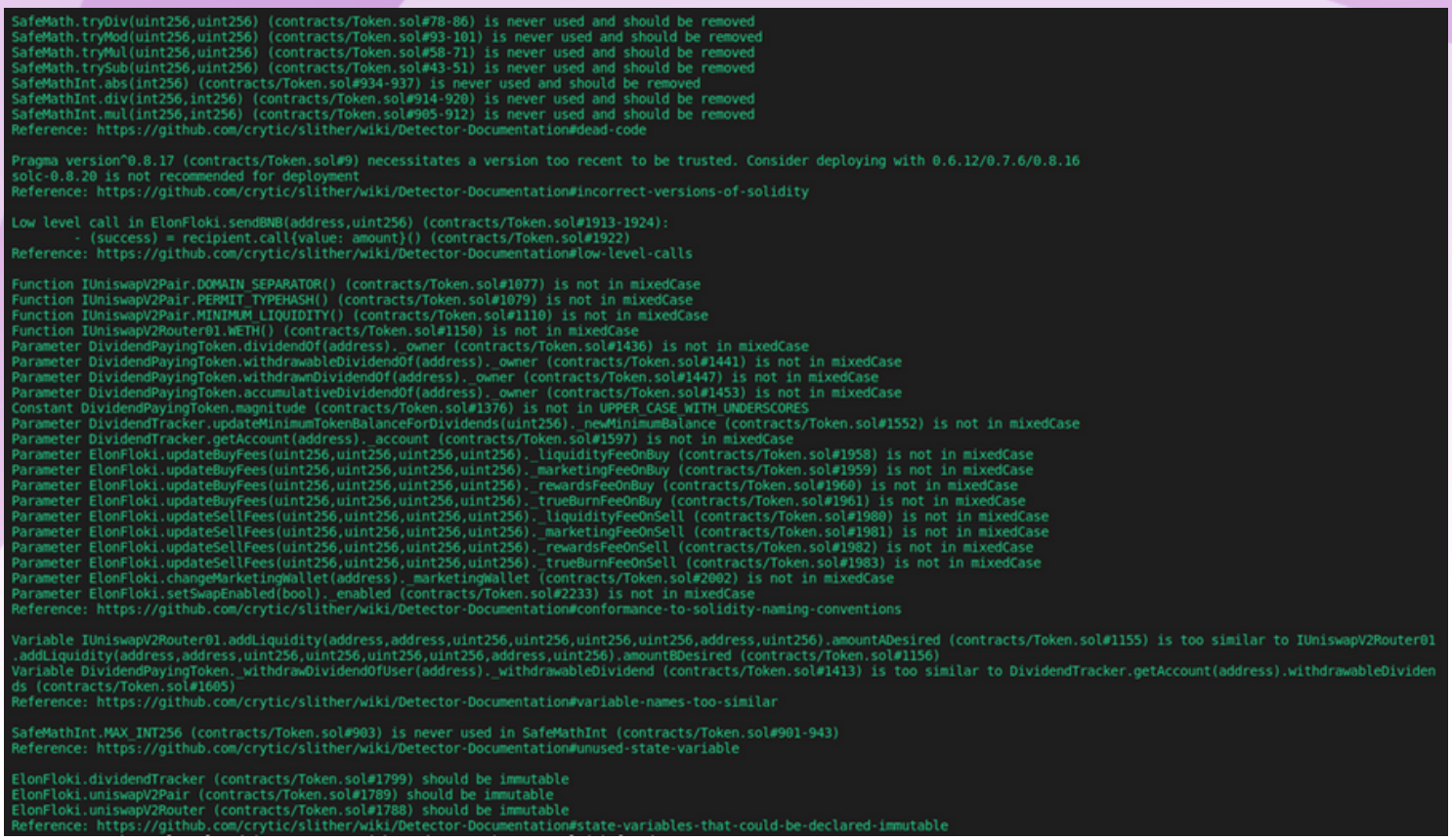
INHERITANCE TREE



POINTS TO NOTE

- **Owner is not able to change buy/sell/transfer fees over 10% each**
- Owner is not able to blacklist an arbitrary address.
- Owner is not able to disable trades
- Owner is not able to set max buy/sell/transfer/hold amount to 0
- Owner is not able to mint new tokens
- **Owner must enable trades manually**

STATIC ANALYSIS



```
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#78-86) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#93-101) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#58-71) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#43-51) is never used and should be removed
SafeMathInt.abs(int256) (contracts/Token.sol#934-937) is never used and should be removed
SafeMathInt.div(int256,int256) (contracts/Token.sol#914-920) is never used and should be removed
SafeMathInt.mul(int256,int256) (contracts/Token.sol#905-912) 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#9) 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

Low level call in ElonFloki.sendBNB(address,uint256) (contracts/Token.sol#1913-1924):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#1922)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#1077) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#1079) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#1110) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/Token.sol#1150) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (contracts/Token.sol#1436) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (contracts/Token.sol#1441) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (contracts/Token.sol#1447) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (contracts/Token.sol#1453) is not in mixedCase
Constant DividendPayingToken.magnitude (contracts/Token.sol#1376) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.updateMinimumTokenBalanceForDividends(uint256)._newMinimumBalance (contracts/Token.sol#1552) is not in mixedCase
Parameter DividendTracker.getAccount(address)._account (contracts/Token.sol#1597) is not in mixedCase
Parameter ElonFloki.updateBuyFees(uint256,uint256,uint256,uint256)._liquidityFeeOnBuy (contracts/Token.sol#1958) is not in mixedCase
Parameter ElonFloki.updateBuyFees(uint256,uint256,uint256,uint256)._marketingFeeOnBuy (contracts/Token.sol#1959) is not in mixedCase
Parameter ElonFloki.updateBuyFees(uint256,uint256,uint256,uint256)._rewardsFeeOnBuy (contracts/Token.sol#1960) is not in mixedCase
Parameter ElonFloki.updateBuyFees(uint256,uint256,uint256,uint256)._trueBurnFeeOnBuy (contracts/Token.sol#1961) is not in mixedCase
Parameter ElonFloki.updateSellFees(uint256,uint256,uint256,uint256)._liquidityFeeOnSell (contracts/Token.sol#1980) is not in mixedCase
Parameter ElonFloki.updateSellFees(uint256,uint256,uint256,uint256)._marketingFeeOnSell (contracts/Token.sol#1981) is not in mixedCase
Parameter ElonFloki.updateSellFees(uint256,uint256,uint256,uint256)._rewardsFeeOnSell (contracts/Token.sol#1982) is not in mixedCase
Parameter ElonFloki.updateSellFees(uint256,uint256,uint256,uint256)._trueBurnFeeOnSell (contracts/Token.sol#1983) is not in mixedCase
Parameter ElonFloki.changeMarketingWallet(address)._marketingWallet (contracts/Token.sol#2002) is not in mixedCase
Parameter ElonFloki.setSwapEnabled(bool)._enabled (contracts/Token.sol#2233) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#1155) is too similar to IUniswapV2Router01
.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#1156)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (contracts/Token.sol#1413) is too similar to DividendTracker.getAccount(address).withdrawableDividen
ds (contracts/Token.sol#1605)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

SafeMathInt.MAX_INT256 (contracts/Token.sol#903) is never used in SafeMathInt (contracts/Token.sol#901-943)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

ElonFloki.dividendTracker (contracts/Token.sol#1799) should be immutable
ElonFloki.uniswapV2Pair (contracts/Token.sol#1789) should be immutable
ElonFloki.uniswapV2Router (contracts/Token.sol#1788) 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 by owner (trades disabled) (passed):
https://testnet.bscscan.com/tx/0xe034393382e7b67d571b54e212456f7649551535da64f9f8cce52d6b6cb1ea16

2- Buying when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0xc93bb46d1e5eff24e35319a24f97af9bfe470dac9c774e25fb07f2ac87874e45

3- Selling when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0xe3ea1008e3480be7ce008e648b62dd3c1d6c961a5597668897a5205500edd75d

4- Transferring when excluded from fees before enabling trades (0% tax) (passed):
https://testnet.bscscan.com/tx/0x60e07173e9d793fa9ceda326d760e02fa88d37d1282b20e480cb97ad29ce083c

5- Buying when not excluded from fees (0-10% tax) (passed):
https://testnet.bscscan.com/tx/0xfc30a4a5318d2b8bc425d22a7c7b06d1b9340bef81dfbb80089cbaf98bd9d6ae

FUNCTIONAL TESTING

6- Selling when not excluded from fees (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0xc92f551a22aa9387aabe5d65f1c04f29d76f3d13ddb661fa7d7c05bcc6957d44

7- Transferring when not excluded from fees (0-10% tax) (passed):

https://testnet.bscscan.com/tx/0x9b5f0f3778e9c4e4518ce099cdcbbbfa0da23915a20394128f037b316087cec1

8- Internal swap & rewards distribution (passed):

- BNB sent to marketing wallet
- A portion of tokens burnt
- A portion of tokens were added to liquidity
- A portion of tokens swapped to BUSD and sent to dividend tracker
- Dividend tracker distributed reward tokens (BUSD)

https://testnet.bscscan.com/tx/0xc92f551a22aa9387aabe5d65f1c04f29d76f3d13ddb661fa7d7c05bcc6957d44

MANUAL TESTING

Logical – Invalid decimal

Severity: Critical

Status: Not Resolved

Overview:

ERC20 contract which is inherited by ElonFoki token and dividend tracker, is returning “18” as decimal number. While other contracts are using “9”.

Suggestion

ERC20 must return “9” as decimal number:

```
function decimals() public view virtual override returns (uint8) {
    return 9;
}
```

MANUAL TESTING

Centralization – Trades must be enabled

Severity: High

function: enableTrading

Status: Not Resolved

Overview:

The smart contract owner must enable trades for holders. If trading remain disabled, no one would be able to buy/sell/transfer tokens.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
}
```

Suggestion

To mitigate this centralization issue, we propose the following options:

1. Renounce Ownership: Consider relinquishing control of the smart contract by renouncing ownership. This would remove the ability for a single entity to manipulate the router, reducing centralization risks.
2. Multi-signature Wallet: Transfer ownership to a multi-signature wallet. This would require multiple approvals for any changes to the mainRouter, adding an additional layer of security and reducing the centralization risk.
3. Transfer ownership to a trusted and valid 3rd party in order to guarantee enabling of the trades

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