



# Smart Contract Audit

FOR  
**Zeldalnu**

DATED : 27 October 23'



# AUDIT SUMMARY

---

**Project name –** Zeldalnu

**Date:** 27 October 2023

**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	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/0x3BeADe8953289D9932A418ff42c59185F3dBCe53>

---



# Token Information

---

**Token Address :**

0xadB6CB9550652739c08Cf890Fb135500a9a756a3

**Name:** Zeldalnu

**Symbol:** ZELDA

**Decimals:** 18

**Network:** Ethereum

**Token Type:** ERC20

**Owner:** 0x98C523f5bFfe44aA8b94A87A83c2721A1cb2DCbc

**Deployer:**

0x98C523f5bFfe44aA8b94A87A83c2721A1cb2DCbc

**Token Supply:** 420, 000, 000, 000, 000, 000

**Checksum:**

1666029b29a5f1ae543a23971ebc1e066fc0f1b5

**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/0x3BeADe8953289D9932A418ff42c59185F3dBCe53>

---



# TOKEN OVERVIEW

---

**buy fee:** 5%

---

**Sell fee:** 12%

---

**transfer fee:** 5%

---

**Fee Privilege:** Owner

---

**Ownership:** Owned

---

**Minting:** None

---

**Max Tx:** No

---

**Blacklist:** No

---

**Other Privileges:**

- Initial distribution of the tokens

---



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

0

### ◆ Medium-Risk

0

### ◆ Low-Risk

0

### ◆ Gas Optimization / Suggestions

0





# INHERITANCE TREE

---





# POINTS TO NOTE

---

- **Owner is not able to change current fees (5% buys and 12% sells)**
  - Owner is not able to blacklist an arbitrary wallet
  - Owner is not able to disable trades
  - Owner is not able to mint new tokens
  - **Owner is not able to set maxTx and max wallet**
-

# STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in TokenDividend.manualSend() (contracts/Token.sol#507-514):
  External calls:
    - address(marketingFeeReceiver).transfer(address(this).balance) (contracts/Token.sol#508)
  State variables written after the call(s):
    - _basicTransfer(address(this),marketingFeeReceiver,balanceOf(address(this))) (contracts/Token.sol#509-513)
      - _balances[sender] = _balances[sender] - amount (contracts/Token.sol#412)
      - _balances[recipient] = _balances[recipient] + amount (contracts/Token.sol#413)
  Event emitted after the call(s):
    - Transfer(sender,recipient,amount) (contracts/Token.sol#414)
    - _basicTransfer(address(this),marketingFeeReceiver,balanceOf(address(this))) (contracts/Token.sol#509-513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
TokenDividend.constructor() (contracts/Token.sol#155-177) uses literals with too many digits:
  - _allowances[address(this)][address(router)] = 100000000 * (10 ** 50) * 100 (contracts/Token.sol#157-160)
TokenDividend.setDistributorSettings(uint256) (contracts/Token.sol#533-536) uses literals with too many digits:
  - require(bool)(gas < 3000000) (contracts/Token.sol#534)
TokenDividend.setTXBNBgas(uint256) (contracts/Token.sol#538-541) uses literals with too many digits:
  - require(bool)(gas < 100000) (contracts/Token.sol#539)
TokenDividend.setDistributorBuyGas(uint256) (contracts/Token.sol#543-546) uses literals with too many digits:
  - require(bool)(gas < 1000000) (contracts/Token.sol#544)
TokenDividend.setLiquidifyGas(uint256) (contracts/Token.sol#548-551) uses literals with too many digits:
  - require(bool)(gas < 1000000) (contracts/Token.sol#549)
TokenDividend.slitherConstructorVariables() (contracts/Token.sol#97-581) uses literals with too many digits:
  - distributorGas = 300000 (contracts/Token.sol#130)
TokenDividend.slitherConstructorVariables() (contracts/Token.sol#97-581) uses literals with too many digits:
  - distributorBuyGas = 400000 (contracts/Token.sol#132)
TokenDividend.slitherConstructorVariables() (contracts/Token.sol#97-581) uses literals with too many digits:
  - LiquidifyGas = 500000 (contracts/Token.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
TokenDividend._decimals (contracts/Token.sol#105) should be constant
TokenDividend._name (contracts/Token.sol#103) should be constant
TokenDividend._symbol (contracts/Token.sol#104) should be constant
TokenDividend.burnFee (contracts/Token.sol#121) should be constant
TokenDividend.buyTax (contracts/Token.sol#114) should be constant
TokenDividend.distributorFactory (contracts/Token.sol#98) should be constant
TokenDividend.feeDenominator (contracts/Token.sol#129) should be constant
TokenDividend.liquidityFee (contracts/Token.sol#118) should be constant
TokenDividend.marketingFee (contracts/Token.sol#119) should be constant
TokenDividend.reflectionFee (contracts/Token.sol#120) should be constant
TokenDividend.router (contracts/Token.sol#100) should be constant
TokenDividend.sellBurnFee (contracts/Token.sol#127) should be constant
TokenDividend.sellLiquidityFee (contracts/Token.sol#124) should be constant
TokenDividend.sellMarketingFee (contracts/Token.sol#125) should be constant
TokenDividend.sellReflectionFee (contracts/Token.sol#126) should be constant
TokenDividend.sellTax (contracts/Token.sol#115) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
TokenDividend.WBNB (contracts/Token.sol#101) should be immutable
TokenDividend.distributor (contracts/Token.sol#99) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:./contracts/Token.sol analyzed (6 contracts with 88 detectors), 70 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**



# CONTRACT ASSESMENT

```
| Contract|      Type      |Bases |      |      |
|:-----:|:-----:|:-----:|:-----:|:-----:|
|  └─ **Function Name** |**Visibility** | **Mutability** |**Modifiers** |
|||||
| **IDEXFactory** | Interface | |||
|  └─ createPair | External  ! | ● |NO ! |
|||||
| **BEP20** | Interface | |||
|  └─ balanceOf | External  ! | |NO ! |
|  └─ transfer | External  ! | ● |NO ! |
|  └─ approve | External  ! | ● |NO ! |
|  └─ transferFrom | External  ! | ● |NO ! |
|||||
| **IDEXRouter** | Interface | |||
|  └─ factory | External  ! | |NO ! |
|  └─ WETH | External  ! | |NO ! |
|  └─ addLiquidityETH | External  ! | 🟢 |NO ! |
|  └─ swapExactTokensForETHSupportingFeeOnTransferTokens | External  ! | ● |NO ! |
|  └─ swapExactETHForTokensSupportingFeeOnTransferTokens | External  ! | 🟢 |NO ! |
|  └─ getAmountsOut | External  ! | |NO ! |
|||||
| **IDividendDistributor** | Interface ||||
|  └─ setDistributionCriteria | External  ! | ● |NO ! |
|  └─ setShare | External  ! | ● |NO ! |
|  └─ deposit | External  ! | 🟢 |NO ! |
|  └─ process | External  ! | ● |NO ! |
|  └─ claimDividend | External  ! | ● |NO ! |
|  └─ getUnpaidEarnings | External  ! | |NO ! |
|||||
| **IDividendDistributorFactory** | Interface | |||
|  └─ createDistributor | External  ! | ● |NO ! |
|||||
```



# CONTRACT ASSESMENT

```
| **TokenDividend** | Implementation | |||
|  ↳ | <Constructor> | Public ! | 🟢 | NO ! |
|  ↳ | <Receive Ether> | External ! | 🟢 | NO ! |
|  ↳ | totalSupply | External ! | | NO ! |
|  ↳ | owner | Public ! | | NO ! |
|  ↳ | decimals | External ! | | NO ! |
|  ↳ | symbol | External ! | | NO ! |
|  ↳ | name | External ! | | NO ! |
|  ↳ | getOwner | External ! | | NO ! |
|  ↳ | balanceOf | Public ! | | NO ! |
|  ↳ | allowance | External ! | | NO ! |
|  ↳ | transfer | External ! | 🔴 | NO ! |
|  ↳ | approve | Public ! | 🔴 | NO ! |
|  ↳ | transferFrom | External ! | 🔴 | NO ! |
|  ↳ | setPair | Public ! | 🔴 | onlyOwner |
|  ↳ | excludeFromFee | Public ! | 🔴 | onlyOwner |
|  ↳ | includeInFee | Public ! | 🔴 | onlyOwner |
|  ↳ | setDividendExempt | Public ! | 🔴 | onlyOwner |
|  ↳ | isExcludedFromFee | Public ! | | NO ! |
|  ↳ | burn | External ! | 🔴 | NO ! |
|  ↳ | _burn | Internal 🔒 | 🔴 | |
|  ↳ | _burnIN | Internal 🔒 | 🔴 | |
|  ↳ | shouldSwapBack | Internal 🔒 | | |
|  ↳ | setmarketingFeeReceivers | External ! | 🔴 | onlyOwner |
|  ↳ | setbuytokensReceiver | External ! | 🔴 | onlyOwner |
|  ↳ | setSwapBackSettings | External ! | 🔴 | onlyOwner |
|  ↳ | value | Public ! | | NO ! |
|  ↳ | _isSell | Internal 🔒 | | |
|  ↳ | BURNFEE | Internal 🔒 | | |
|  ↳ | MKTFEE | Internal 🔒 | | |
|  ↳ | LIQUIFYFEE | Internal 🔒 | | |
|  ↳ | REFFEE | Internal 🔒 | | |
|  ↳ | _transferFrom | Internal 🔒 | 🔴 | |
|  ↳ | _basicTransfer | Internal 🔒 | 🔴 | |
|  ↳ | _txTransfer | Internal 🔒 | 🔴 | |
|  ↳ | getamount | Internal 🔒 | | |
|  ↳ | swapBack | Internal 🔒 | 🔴 | swapping |
```



# CONTRACT ASSESMENT

---

	manualSend	External !		onlyOwner
	setDistributionCriteria	External !		onlyOwner
	claimDividend	External !		NO !
	getUnpaidEarnings	Public !		NO !
	setDistributorSettings	External !		onlyOwner
	setTXBNBgas	External !		onlyOwner
	setDistributorBuyGas	External !		onlyOwner
	setLiquidifyGas	External !		onlyOwner
	renounceOwnership	Public !		onlyOwner
	transferOwnership	Public !		onlyOwner
	\_transferOwnership	Internal		

## ### Legend

Symbol	Meaning
	Function can modify state
	Function is payable



# FUNCTIONAL TESTING

---

## 1- Adding liquidity (**passed**):

<https://testnet.bscscan.com/tx/0x59758d5a903e144ab53ac50d745eccf077e772b2e55b217ddfd74a15699825f3>

## 2- Buying when excluded (0% tax) (**passed**):

<https://testnet.bscscan.com/tx/0x46646d452ee62faac20f3bd026b42e108921a1aa4028fd53b034563022dd70c3>

## 3- Selling when excluded (0% tax) (**passed**):

<https://testnet.bscscan.com/tx/0xe11b99f63901df12b361650293d983d827cde21ae26f05939715dd3f298ac421>

## 4- Transferring when excluded from fees (0% tax) (**passed**):

<https://testnet.bscscan.com/tx/0x6ca9d3c66938b013341b8b51d77f7e832b99c18b16560c42a5beb56281e4c33d>

## 5- Buying when not excluded from fees (tax 5%) (**passed**):

<https://testnet.bscscan.com/tx/0xa42ceda036298a0c96ce7b930cefe8087541286de5c78fa040e90335fe4db841>

## 6- Selling when not excluded from fees (tax 0-12%) (**passed**):

<https://testnet.bscscan.com/tx/0x357eb59930ef3589f006b77a02ac1e2f8a3014dfdc464cdd2ded9af5cb6a7ba4>

## 7- Transferring when not excluded from fees (5 % tax ) (**passed**):

<https://testnet.bscscan.com/tx/0x2b39b55999a0223d3f5edd3ff4977116f7e116e05bd3827b02caccb57a8bbcb0>

## 7- Internal swap (Auto-liquidity + Marketing fee + reflections) (**passed**):

<https://testnet.bscscan.com/tx/0x357eb59930ef3589f006b77a02ac1e2f8a3014dfdc464cdd2ded9af5cb6a7ba4>

---



# 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 specialize 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://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---