PROOF

ZELDA INU

SMART CONTRACT

SECURITY AUDIT REPORT

Disclaimer

This is a limited report of findings based on an analysis of industry best practices as of the date of this report regarding cybersecurity vulnerabilities and issues in smart contract frameworks and algorithms, the details of which are detailed in this report. stated in the report. To get the full picture of our analysis, it's important to read the full report. Although we have conducted our analysis and have done our best to prepare this report, you should not rely on this report and cannot claim against us based on what it does or does not say or how it was produced. It is important to do your own research before making any decisions. This is explained in more detail in the following disclaimer. Please be sure to read to the end.

Disclaimer:

BY READING THIS REPORT OR ANY PART THEREOF, YOU AGREE TO THE TERMS OF THIS DISCLAIMER. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Proof Audit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Proof Audit) owe no duty of care towards you or any other person, nor does Proof Audit make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Proof Audit hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Proof Audit hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Proof Audit, for any amount or type of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

Security analytics are based solely on smart contracts. Application or process security not checked. Product code not reviewed.

Table of Contents

Executive Summary

Objectives

Proof Audit, carried out an audit of Zelda Inu, specifically their BEP20 token. The project is based on the BNB Chain Network. We reviewed documentation which helped with understanding the functions of their code. Our findings in the audit ranged from minor to critical.

Project Info

Audited project

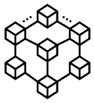
Zelda Inu

Contacts

N/A

Deployer Address

0x26a7546c8f5e5f706cb598CAA68134f6eCf8d657



Blockchain

BNB Chain

Project website:

https://zeldainu.com/

Methodology

During the audit process, we inspected the repository thoroughly, using a line-by-line code read through to review vulnerabilities, quality of the code and adherence to best practices and specifications. We used Computer-Aided Verification to support the audit process.

Our auditing process is as follows:

1. **Code Review:**
   A review of the scope, specifications and documentation provided to ensure an in depth understanding of the purpose and functionality of the relevant smart contracts.
2. **Automated Analysis:**
   A series of reviews carried out with the use of automated tools. These reviews serve as a basis for further manual analysis and provide relevant visualizations of the code.
3. **Testing & Manual Review of Code:**
   Test coverage analysis and a line-by-line read through of the project code in order to identify vulnerabilities, errors and weaknesses in code quality.
4. **Specification Comparison:**
   A review of the code against the specifications provided to ensure that the code operates as is intended.
5. **Best Practices Review:**
   A review of the smart contracts to identify potential improvements in effectiveness, efficiency, and maintainability, with a focus on adherence to industry best practices.

Scope

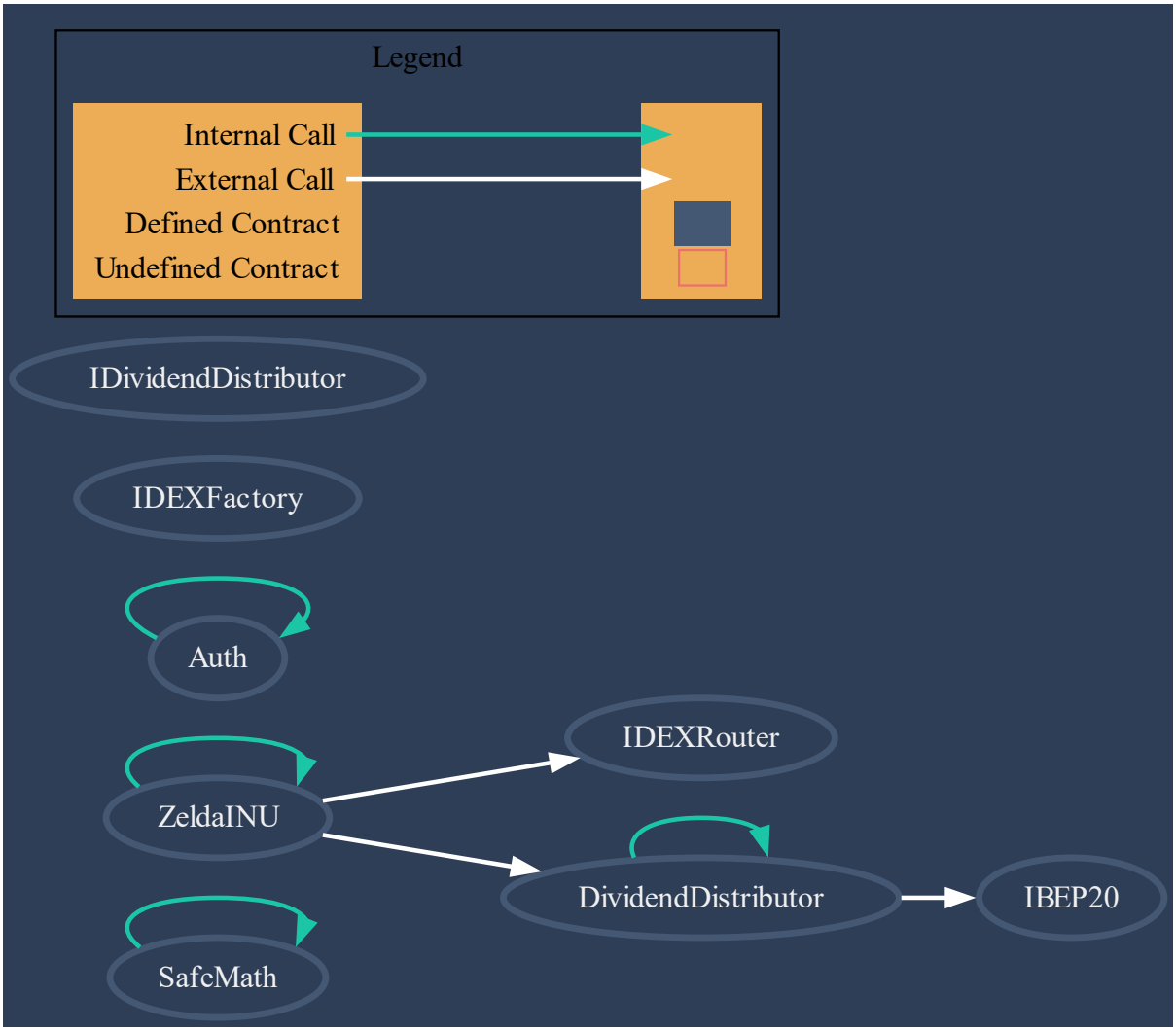
The contracts audited are from BscScan at:
https://bscscan.com/address/0x1E3220194F8f65646418b44da87F41F073b24a27#code.

The audited contracts are:

ZeldaInu.sol

The scope of the audit is limited to these files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

ZeldaInu.sol Interaction Graph



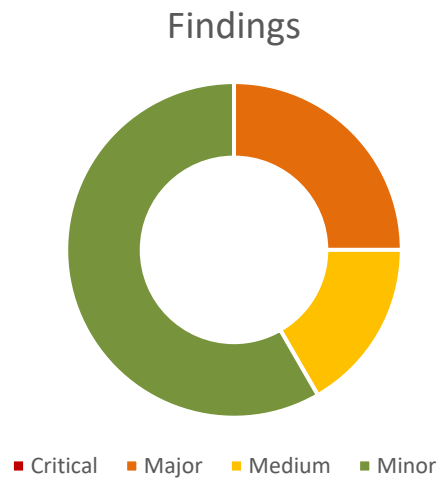
Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors
- Outdated version of Solidity compiler
- Race conditions
- Reentrancy attacks
- Misuse of block timestamps
- Denial of service attacks
- Excessive gas usage
- Missing or misused function qualifiers
- Needlessly complex code and contract interactions
- Poor or nonexistent error handling
- Insufficient validation of the input parameters
- Incorrect handling of cryptographic signatures
- Centralization and upgradeability

Summary of Findings

We found **0** critical issue, **3** Major issues, **2** medium issues and **7** minor issues.



Security Issues

ID	Title	Severity	Status
01	Insufficient Gas	Major	**Acknowledged**
02	Minting to Centralized Address	Major	**Acknowledged**
03	Centralization Risk	Major	**Acknowledged**
04	Financial Models	Medium	**Acknowledged**
05	Upper Limit for Total Fees is notReasonable	Medium	**Acknowledged**
06	Third Party Dependencies	Minor	**Acknowledged**
07	Check Effect Interaction Pattern Violated	Minor	**Acknowledged**
09	Potential Sandwich Attacks	Minor	**Acknowledged**
10	set_sell_multiplier LacksRestrictions	Minor	**Acknowledged**

11	Miscalculation in swapBack()	Minor	**Acknowledged**
12	Miscalculation of Max Holding	Minor	**Acknowledged**
13	Permission Not Removed	Minor	**Acknowledged**

Findings

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them, or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Partially resolved**: Adjusted program implementation to eliminate part of the risk. The other part remains in the code but is a result of an intentional decision.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk.

Critical Severity Issues

N/A

Major Severity Issues

Insufficient Gas

Description: In the process() function, there may be insufficient gas during the last dividend distribution process,causing the function to roll back.

Recommendation: We recommend using the claimDividend() function to allow users to manually claim dividends.

Status: **Acknowledged**

Minting To Centralized Address

Description: The full amount of totalSupply tokens are initially minted to the msg.sender address belonging to the contract owner.

Recommendation: The private keys of the owner accounts should be carefully protected to avoid potential risks of hacking.

Status: **Acknowledged**

Centralization Risk

Description: In the contract ZeldaInu , the role owner has the authority over the following functions:

- setMaxWalletPercent_base1000()
- setMaxTxPercent_base1000()
- set_sell_multiplier()
- tradingStatus()
- enable_blacklist()
- manage_blacklist()
- multiTransfer()
- multiTransfer_fixed()

Recommendation: The current project design involves some risks, which can be mitigated by making improvements to security and decentralization. However, these issues cannot be fully resolved at this time. We advise the client to be cautious with their privileged account's private key to prevent hacking. We also suggest that the use of centralized privileges or roles in the protocol be improved by using

decentralized mechanisms or smart contracts with enhanced security measures such as multi-signature wallets.

Status: **Acknowledged**

Medium Severity Issues

Financial Models

Description: The Zelda Inu protocol is a DeFi token on the Binance smart chain. If neither the sender nor receiver is exempt from fees, fees will be charged for the transaction, including liquidity fees, reflection fees, marketing fees, ecosystem fees, and burn fees. The owner can change the fee rates at any time. If the transaction is a sell, the fees may be adjusted using a sell multiplier. The burn fee is sent directly to the burn fee receiver, while the other fees are sent to the contract address first. If certain conditions are met, these fees will be used to add liquidity and be converted to BNB. The LP tokens are for the auto liquidity receiver, and a portion of the converted BNB is sent to the marketing fee receiver and ecosystem fee receiver. The remaining BNB is deposited into the dividend distributor contract, where it is used to swap reward tokens for dividend holders. Holders can claim their rewards on their own or wait for automatic distribution triggered by each transaction.

Recommendation: We recommend that the client should publicly display the financial models for their community to see.

Status: **Acknowledged**

Upper Limit For Total Fees Is Not Reasonable

Description: The current maximum transaction fee is set at 50%, which is not a reasonable value.

Recommendation: We recommend adding an upper limit for total fee and set it to an appropriate value such as 15%.

Status: **Acknowledged**

Minor Severity Issues

Third Party Dependencies

Description: The contract interacts with the Uniswap third party protocol.
Third party protocols are not covered in the scope of the audit. There is a potential for third party protocols to be compromised.

Recommendation: Interaction with third party dependencies is necessary for the CliffordInu functionality, therefore we recommend that the team does their due diligence to ensure the protocols they are interacting with are secure and remain secure.

Status: **Acknowledged**

Check Effect Interaction Pattern Violated

Description: A reentrancy attack happens when a contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would not have happened after the external call resolved the effects.

Recommendation: We recommend the client to adhere to the Check Effect Interaction Pattern.

Status: **Acknowledged**

Potential Sandwich Attacks

Description: Sandwich attacks occur when an attacker places an order directly before a victim transaction and also places one directly after it. In essence, the attacker will front-run and back-run simultaneously, with the original pending transaction sandwiched in between.

The deposit() and swapBack() functions are called with no restriction on minimum output amount or slippage, making them vulnerable to sandwich attacks.

Recommendation: We recommend setting minimum output amounts greater than 0 for the relevant functions.

Status: **Acknowledged**

set_sell_multiplier Lacks Restrictions

Description: If both sender and recipient are not in isFeeExempt , the transaction will charge a certain fee.
Since the maximum value of totalFee is 50% of feeDenominator , and the sellMultiplier does not setrestrictions, theoretically, when the multiplier is set to 200, all the transaction amount will be deducted as tax.

Recommendation: We recommend adding the restriction for the sellMultiplier to avoid unexpected losses caused by misoperation or leakage of the owner account private key.

Status: **Acknowledged**

Miscalculation In swapBack()

Description: The totalFee includes the burnFee . However, the burn fee is sent to burnFeeReceiver directly. It is wrong to use the totalFee as the fee denominator in swapBack . And in this wrong way, there will be someBNB left in the contract. The correct way is using totalFee - burnFee as the fee denominator.

Recommendation: We recommend the client considers uses the more accurate method.

Status: **Acknowledged**

Miscalculation Of Max Holding

Description: The transaction may be charged fees, so the max holding of receiver should be `heldTokens` +`amountReceived` . The fees should not be calculated in the max holding.

Recommendation: We recommend the client checking max holding after calculating the `amountReceived` .

Status: **Acknowledged**

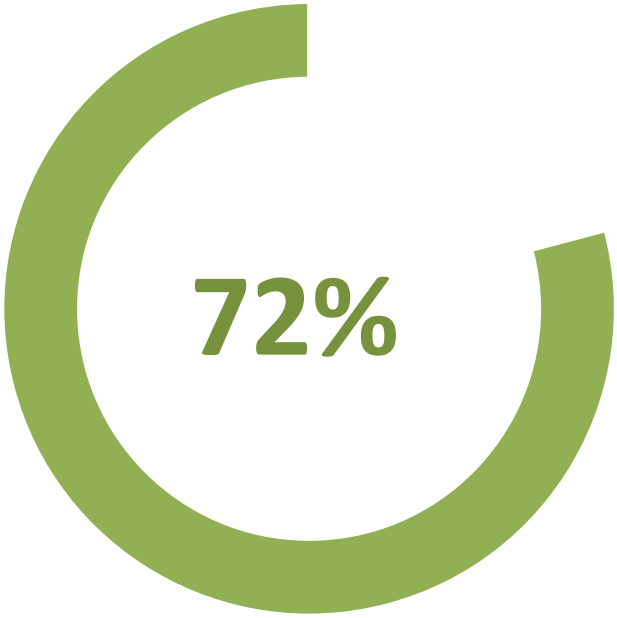
Permission Not Removed

Description: After `transferOwnership()` transfers the authority, the original owner still has the `authorized` authority.

Recommendation: We recommend for the client to remove the authority of original owner when it transfers authority.

Status: **Acknowledged**

Security Rating



Based on Vulnerabilities Found