



# Security Assessment ZUDO FINANCE

Vital Block Verified on April 16<sup>th</sup>, 2023

@Vital-Block

@VB\_Audit

info@vitalblock.org



www.vitalblock.org



PREPARED FOR:  
ZUDO FINANCE



## INTRODUCTION

<b>Auditing Firm</b>	 <b>VITAL BLOCK SECURITY</b>
<b>Client Firm</b>	 <b>ZUDO FINANCE</b>
<b>Methodology</b>	Automated Analysis, Manual Code Review
<b>Language</b>	Solidity
<b>Contract</b>	<p><b>ZUDO Token:</b> 0x4D82DBE88E8cF07e27b9235aB21924F33bB98D91</p> <p><b>Factory:</b> 0x67eb631Bda782188949ca1e15d2e778adb227FB1</p> <p><b>Router:</b> 0xc3bbfae50DFdD6625FEf623Ace912a514097aD95</p> <p><b>Masterchef:</b> 0x1129F03471Efa75c8332cA6699beeAC816DE70e4</p> <p><b>Presale:</b> 0x91cBeD889B58f6B9Ff35c4c4792A2690248fa014</p>
<b>Blockchain</b>	Zksync
<b>Centralization</b>	Active ownership
<b>Website</b>	<a href="https://zudo.finance/">https://zudo.finance/</a>
<b>Discord</b>	<a href="https://discord.com/invite/mmWAcbjv6C">https://discord.com/invite/mmWAcbjv6C</a>
<b>Twitter</b>	<a href="https://twitter.com/zudo_finance">https://twitter.com/zudo_finance</a>
<b>Medium</b>	<a href="https://medium.com/@ZUDO_Finance">https://medium.com/@ZUDO_Finance</a>
<b>GitHub</b>	<a href="https://github.com/zudo-finance/zudo-core">https://github.com/zudo-finance/zudo-core</a>
<b>Litepaper</b>	<a href="https://zudo-finance.gitbook.io/">https://zudo-finance.gitbook.io/</a>
<b>Prelim Report Date</b>	April 14, 2023
<b>Final Report Date</b>	April 16 2023



Verify the authenticity of this report on our GitHub Repo: <https://www.github.com/vital-block>



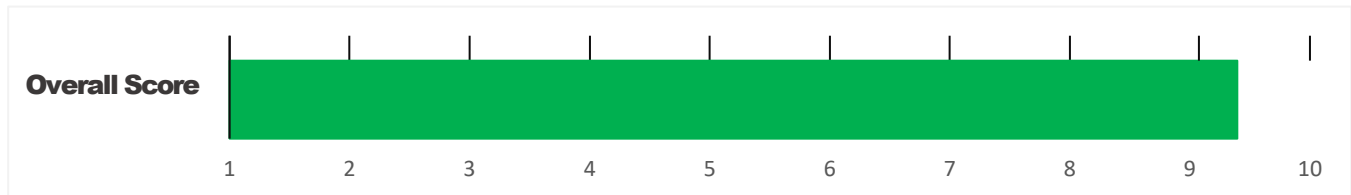
## EXECUTIVE SUMMARY


**ZUDO FINANCE** has performed the automated and manual analysis of the **ZUDO FINANCE** Sol code. The code was

reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

Status	Critical ! 🔴	Major " 🟡	Medium # 🟡	Minor \$ 🟢	Unknown % 🟤
Open	0	0	1	2	0
Acknowledged	0	0	1	3	0
Resolved	0	0	0	0	0
Noteworthy <a href="#">onlyOwner</a> Privileges	Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router				

**ZUDO FINANCE** Smart contract has achieved the following score: **94.5**



 Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

 Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.



# TABLE OF CONTENTS

TABLE OF CONTENTS .....	4
SCOPE OF WORK .....	5
AUDIT METHODOLOGY .....	6
RISK CATEGORIES .....	8
CENTRALIZED PRIVILEGES .....	9
AUTOMATED ANALYSIS .....	10
INHERITANCE GRAPH .....	15
MANUAL REVIEW .....	16
DISCLAIMERS .....	27
ABOUT VITALBLOCK .....	30



## SCOPE OF WORK

Vital Block was consulted by ZUDO FINANCE to conduct the smart contract audit of its. Sol source code. The audit scope of work is strictly limited to mentioned .SOL file only:

- ZUDOFINANCE.Sol

 External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

Public Contract Link	
<b>0x4D82DBE88E8cF07e27b9235aB21924F33bB98D91</b>	
Contract Name	ZUDO FINANCE
Token Symbol	ZUDO
Decimals	18
Total Supply	21,000,000

## AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block auditing process and methodology:

### CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

### AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges.

We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

Centralized Exploits	<ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul>
----------------------	---



### **Common Contract Vulnerabilities**

- **Integer Overflow**
- **Lack of Arbitrary limits**
- **Incorrect Inheritance Order**
- **Typographical Errors**
- **Requirement Violation**
- **Gas Optimization**
- **Coding Style Violations**
- **Re-entrancy**
- **Third-Party Dependencies**
- **Potential Sandwich Attacks**
- **Irrelevant Codes**
- **Divide before multiply**
- **Conformance to Solidity Naming Guides**
- **Compiler Specific Warnings**
- **Language Specific Warnings**

### **REPORT**

- **The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.**
- **The client's development team reviews the report and makes amendments to the codes.**
- **The auditing team provides the final comprehensive report with open and unresolved issues.**

### **PUBLISH**

- **The client may use the audit report internally or disclose it publicly.**






 **It is important to note that there is no pass or fail in the audit, it is recommended to view the audit**

**as an unbiased assessment of the safety of solidity codes.**



## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

Risk Type	Definition
<b>Critical</b> ! 	These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
<b>Major</b> " 	These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.
<b>Medium</b> # 	These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits.
<b>Minor</b> \$ 	These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.
<b>Unknown</b> % 	These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
<b>Open</b>	Risks are open.
<b>Acknowledged</b>	Risks are acknowledged, but not fixed.
<b>Resolved</b>	Risks are acknowledged and fixed.





## CENTRALIZED PRIVILEGES

**Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.**

**There are some well-intended reasons have privileged roles, such as:**

- **Privileged roles can be granted the power to `pause()` the contract in case of an external attack.**
- **Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

**Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.**

- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

 **Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.**








# AUDIT SCOPE

## ZUDO.FINANCE

ID	Repo	Comment	File	SHM281 Checksum
ZTM	zudo-finance/zudo-core	cC51D68	ERC20.sol	85f15802c6be0fd50f8632d8433cccc9d b6f4b39f9e566d1fa78de54b84bdd35
ZRY	zudo-finance/zudo-core	cC51D68	ERC20Mock.sol	8oippkjjk96be0fd50f8632d8433cccc9 db6f4b39f9e566d1yhhg8765ffckiuybb
STV	zudo-finance/zudo-core	cC51D68	MasterChef.sol	3666778uj908766362fvyga98jdkl8864 8yhfbqt37409owehbgwhuyyyg223738
ZML	zudo-finance/zudo-core	cC51D68	MerkleProof.sol	98uuyriy399787390uhbiiuhghdg7guu 30oi7799u9359ydfgdgygeigi3ioueyy78
STR	zudo-finance/zudo-core	cC51D68	Presale.sol	4566efgywqtfeuh87872t1537883798 3639293763hhegetgjfwjk89336668862
ZOP	zudo-finance/zudo-core	cC51D68	ZudoToken.sol	546363ttebnve88329973mvvdsaggct47 8153ytdgfdxy792635fgdjgi1900990908
ZDP	Access/zudo-finance/zudo-core	cC51D68	Ownable.sol	835656990327hudbinnjnr6729dchjld0 993ytyy3vq63235727879889073
ZWY	zudo-core/exchange/factory/	cC51D68	ZudoERC20.sol	cc089692343d1cc36eaf196046d7a528 d153abd55ba20e82f1d57c22fcd92675
ZKB	zudo-core/exchange/factory/	cC51D68	ZudoFactory.sol	8448b3af42497f5f74e53424ee3e6c55 1f51356945108d22a893d608a7990542
ZXY	zudo-core/exchange/factory/	cC51D68	ZudoPair.sol	5c86aa1dd3889db5fcd17a80214b226f c784f268ab9db82df97c1d2459467831
ZCB	zudo-core/exchange/router/	cC51D68	ZudoRouter.sol	b8244da33db171e5533d77bef4a3570 3df1de2cebea5f35cb38ce6a26c778cf1
ZWO	zudo-core/token/	cC51D68	SafeERC20.sol	3d408b8f2cc56f9699a402b5151de906 71de089c3007afc9e4fc867c04152e7c
ZGT	zudo-core/utis/	cC51D68	Context.sol	9d751621c3501102e4b50005ca3314ec 6e04e6ff8bbb30852d1c7edfff3f8cef
ZFF	zudo-core/libraries/	cC51D68	SafeMath.sol	455687gfsadjknlpuihhg774580vgfxr ki9876dhgvgb990lkjhde444566788



## AUTOMATED ANALYSIS

Symbol	Definition
	Function modifies state
	Function is payable
	Function is internal
	Function is private
	Function is important

```

**ZUDO FINANCE** | Interface | |||
| L | totalSupply | External | ! | NO |
| L | decimals | External | ! | NO |
| L | symbol | External | ! | NO |
| L | name | External | ! | NO |
| L | getOwner | External | NO |
| L | balanceOf | External | ! | NO |
| L | transfer | External | " ! ! | NO |
| L | allowance | External | ! | NO |
| L | approve | External | " ! ! | NO |
| L | transferFrom | External | " | NO |
|||||
**IFactoryV2** | Interface | |||
| L | getPair | External | NO | |
| L | createPair | External | " | NO |
|||||
**IV2Pair** | Interface | |||
| L | factory | External | NO | |
| L | getReserves | External | NO |
| L | sync | External | " | NO |

```



|||||

| **\*\*IRouter01\*\*** | Interface | |||

| L | factory | External ¶ | |NO¶|

| L | ETH | External ¶ | |NO¶|

| L | addLiquidityETH | External ¶ | # |NO¶|

| L | addLiquidity | External ¶ | " |NO¶|

| L | swapExactAPTFForTokens | External ¶ | # |NO¶|

| L | getAmountsOut | External ¶ | |NO¶|

| L | getAmountsIn | External ¶ | |NO¶|

|||||

| **\*\*IRouter02\*\*** | Interface | IRouter01 |||

| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External ¶ | " |NO¶|

| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External ¶ | # |NO¶|

| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ¶ | " ! 🔴 |NO¶|

| L | swapExactTokensForTokens | External ¶ | " |NO¶|

|||||

| **\*\*Protections\*\*** | Interface | |||

| L | checkUser | External ¶ | " ! 🔴 |NO¶|

| L | setLaunch | External ¶ | " |NO¶|

| L | setLpPair | External ¶ | " |NO¶|

| L | **ZUDO** | External ¶ | " |NO¶|

| L | removeSniper | External ¶ | " |NO¶|

|||||

| **\*\*Cashier\*\*** | Interface | |||

| L | setRewardsProperties | External ¶ | " |NO¶|

| L | tally | External ¶ | " |NO¶|

| L | load | External ¶ | # |NO¶|

| L | cashout | External ¶ | " |NO¶|

| L | giveMeWelfarePlease | External ¶ | " |NO¶|

| L | getTotalDistributed | External ¶ | |NO¶|

| L | getUserInfo | External ¶ | |NO¶|

| L | getUserRealizedRewards | External ¶ | |NO¶|



```

| L | getPendingRewards | External | | | NO |
| L | initialize | External | | " | NO |
| L | getCurrentReward | External | | | NO |
|||||
| **SOL** | Implementation | SafeMath | |||
| L | <Constructor> | Public | | # | NO |
| L | transferOwner | External | | " | onlyOwner |
| L | renounceOwnership | External | | " | NO |
| L | setOperator | Public | | | NO |
| L | renounceOriginalDeployer | External | | " | NO |
| L | <Receive Ether> | External | | # | NO |
| L | totalSupply | External | | | NO |
| L | decimals | External | | | NO |
| L | symbol | External | | | NO |
| L | name | External | | | NO |
| L | getOwner | External | | ! | NO |
| L | balanceOf | Public | | ! | NO |
| L | allowance | External | | ! | NO |
| L | approve | External | | " ! | NO |
| L | _approve | Internal | $ | " | NO |
| L | approveContractContingency | Public | | " ! | onlyOwner |
| L | transfer | External | | | NO |
| L | transferFrom | External | | | NO |
| L | setNewRouter | External | | " | onlyOwner |
| L | setLpPair | External | | | onlyOwner |
| L | setInitializers | External | | " | onlyOwner |
| L | isExcludedFromFees | External | | | NO |
| L | isExcludedFromDividends | External | | | NO |
| L | isExcludedFromProtection | External | | | NO |
| L | setDividendExcluded | Public | | " | onlyOwner |
| L | setExcludedFromFees | Public | | " | onlyOwner |

```



## ZTV-03 POSSIBLE OVERFLOW

Category	Severity <span>●</span>	Location	Status
StatusMathematical Operations	Minor	zudo-core/ZudoToken.sol	Acknowledged

### Description

In `updateForTaker`, the following equation is used inside an unchecked block






```
function safeZudoTransfer(address _to, uint256 _amount) public onlyOwner {  
    uint256 zudoBal = balanceOf(address(this));  
    if (_amount > zudoBal) {
```

Where parameters.addressOutUsed is a `this` and override In is a `this`. As these two are multiplied together in an unchecked block, they may overflow.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the `PairsIn` is close enough it will never cause an overflow

## OPTIMIZATIONS | ZUDO FINANCE

ID	Title	Category	Status
ZTV	Logarithm Refinement Optimization	Gas Optimization	Acknowledged 
ZOP	Checks Can Be Performed Earlier	Gas Optimization	Acknowledged 
ZDP	Unnecessary Use Of SafeMath	Gas Optimization	Acknowledged 
ZWY	Struct Optimization	Gas Optimization	Acknowledged 
ZGT	Unused State Variable	Gas Optimization	Acknowledged 

## General Detectors

### Missing Zero Address Validation

Some functions in this contract may not appropriately check for zero addresses being used.









































Attention  
Required

### Incorrect Solidity Version

This contract uses an unconventional or very old version of Solidity



Attention  
Required

- |  |  |
|--|--|
|  No compiler version inconsistencies found      |  No tautologies or contradictions found                       |
|  No unchecked call responses found              |  No faulty true/false values found                            |
|  No vulnerable self-destruct functions found    |  No innacurate divisions found                                |
|  No assertion vulnerabilities found            |  No redundant constructor calls found                        |
|  No old solidity code found                   |  No vulnerable transfers found                              |
|  No external delegated calls found            |  No vulnerable return values found                          |
|  No external call dependency found            |  No uninitialized local variables found                     |
|  No vulnerable authentication calls found     |  No default function responses found                        |
|  No invalid character typos found             |  No missing arithmetic events found                         |
|  No RTL characters found                      |  No missing access control events found                     |
|  No dead code found                           |  No redundant true/false comparisons found                  |
|  No risky data allocation found               |  No state variables vulnerable through function calls found |
|  No uninitialized state variables found       |  No buggy low-level calls found                             |
|  No uninitialized storage variables found     |  No expensive loops found                                   |
|  No vulnerable initialization functions found |  No bad numeric notation practices found                    |
|  No risky data handling found                 |  No missing constant declarations found                     |
|  No number accuracy bug found                 |  No missing external function declarations found            |
|  No out-of-range number vulnerability found   |  No vulnerable payable functions found                      |
|  No map data deletion vulnerabilities found   |  No vulnerable message values found                         |





## Vulnerability Scan

### REENTRANCY

✓ No reentrancy risk found

Severity Major

Confidence Parameter Certain

✗ **Not Mintable:** A large amount of this token can not be minted by a private wallet or contract.

```
contract ZudoToken is ERC20 {
    using SafeMath for uint256;
    uint256 public constant MAX_TOTAL_SUPPLY = 21_000_000e18;

    constructor(uint256 _initialSupply) ERC20("ZUDO Token",
"ZUDO") {
        _mint(msg.sender, _initialSupply);
    }

    /// @notice Creates `_amount` token to `_to`. Must only
    be called by the owner (MasterChef).
    function mint(address _to, uint256 _amount) public
    onlyOwner {
        require(
            _amount + totalSupply() <= MAX_TOTAL_SUPPLY,
            "ZudoToken: Max total supply exceeded"
        );
        _mint(_to, _amount);
    }
}
```

## Vulnerability Description

## Scanning Line:



## Repository:

<https://github.com/zudo-finance/zudo-core>

## All Audited Files

Zudo.sol  
Factory.sol  
Router.sol  
Masterchef.sol  
Presale.sol

## Contract Creator

0x01B5d5D5b7524E67d3112703822377C8540BB740

## Creator Txn Hash

0x828e99fbc140da76eacf34768032249a824ff7833ec9d2f9345b10b7f2892919

## Contracts:

### Contract :

ZUDO Token: 0x4D82DBE88E8cF07e27b9235aB21924F33bB98D91  
Factory: 0x67eb631Bda782188949ca1e15d2e778adb227FB1  
Router: 0xc3bbfae50DFfD6625FEf623Ace912a514097aD95  
Masterchef: 0x1129F03471Efa75c8332cA6699beeAC816DE70e4  
Presale: 0x91cBeD889B58f6B9Ff35c4c4792A2690248fa014



## Vulnerability Run check

### Risk Analysis

#### ✔ Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

#### ✔ No Proxy

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

#### ✔ No mint function

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

#### ✔ No function to retrieve ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

#### ✔ Owner cant change balance

The contract owner does not have the authority to modify the balance of tokens at other addresses.

### Honeypot Risk

#### ✔ This does not appear to be a honeypot

We are not aware of any code that prevents the sale of tokens.

#### ✔ No trading cooldown

The token contract has no trading cooldown function. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

#### ✔ No Anti Whale

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

#### ✔ No blacklist function

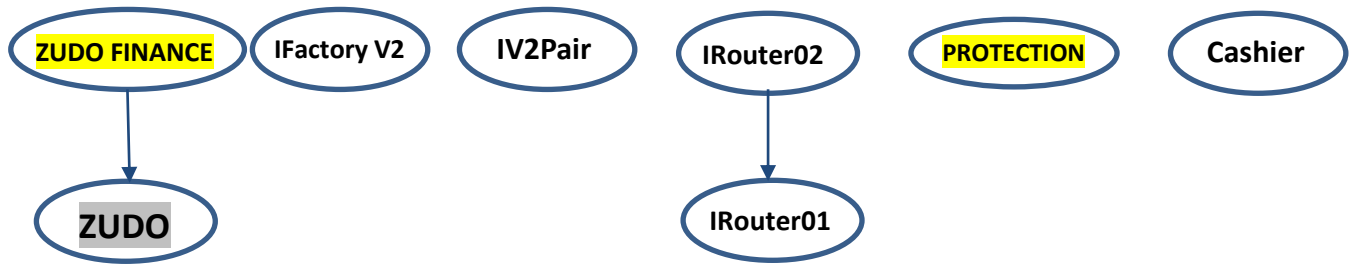
No blacklist function is included.

#### ✔ No whitelist function

Whitelist function found



## INHERITANCE GRAPH



Identifier	Definition	Severity
CEN-12	Centralization privileges of ZUDO FINANCE	Medium # 🟡

Vulnerability 0 : No important security issue detected.

Threat level: Low

```

7  using SafeMath for uint256;
8  uint256 public constant MAX_TOTAL_SUPPLY = 21_000_000e18;
9
10 constructor(uint256 _initialSupply) ERC20("ZUDO Token", "ZUDO") {
11     _mint(msg.sender, _initialSupply);
12 }
13
14 /// @notice Creates `_amount` token to `_to`. Must only be called by the owner (MasterChef).
15 function mint(address _to, uint256 _amount) public onlyOwner {
16     require(
17         _amount + totalSupply() <= MAX_TOTAL_SUPPLY,
18         "ZudoToken: Max total supply exceeded"
19     );
20     _mint(_to, _amount);
21 }
22
23 // Safe zudo transfer function
24 function safeZudoTransfer(address _to, uint256 _amount) public onlyOwner {
25     uint256 zudoBal = balanceOf(address(this));
26     if (_amount > zudoBal) {
27         _transfer(address(this), _to, zudoBal);
28     } else {
29         _transfer(address(this), _to, _amount);
30     }
31 }
  
```

## MANUAL REVIEW

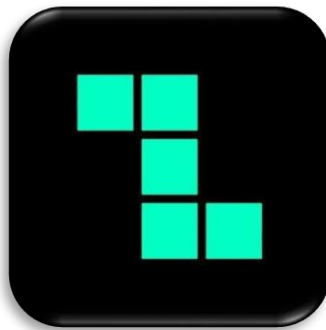
**ZUDO Finance:** is a decentralized exchange (DEX) built on the zkSync Era blockchain. The platform provides users with a seamless trading experience while maintaining high-speed and low-cost transactions.

**TOKEN NAME:** ZUDO FINANCE

**Ticker:** ZUDO

**Chain/Standard:** ZKSYNC

**Total Supply:** 21,000,000



The ZUDO FINANCE Platform Is Launching On Zksync Chain

### Earn with ZUDO

#### WHAT IS ZUDO FINANCE?

ZUDO Finance is a decentralized exchange (DEX) built on the zkSync Era blockchain. The platform provides users with a seamless trading experience while maintaining high-speed and low-cost transactions.

At ZUDO Finance, trading has never been smoother or more rewarding. Thanks to our cutting-edge zero-knowledge technology, we provide users with easy-to-use and low-cost DeFi services without compromising Ethereum's top-notch security.



# ISSUES CHECKING STATUS

Issue Description

Checking Status

1.	Compiler errors.	PASSED
2.	Race Conditions and reentrancy. Cross-Function Race Conditions.	PASSED
3.	Possible Delay In Data Delivery.	PASSED
4.	Oracle calls.	PASSED
5.	Front Running.	PASSED
6.	Sol Dependency.	PASSED
7.	Integer Overflow And Underflow.	PASSED
8.	DoS with Revert.	PASSED
9.	Dos With Block Gas Limit.	PASSED
10.	Methods execution permissions.	PASSED
11.	Economy Model of the contract.	PASSED
12.	The Impact Of Exchange Rate On the solidity Logic.	PASSED
13.	Private use data leaks.	PASSED
14.	Malicious Event log.	PASSED
15.	Scoping and Declarations.	PASSED
16.	Uninitialized storage pointers.	PASSED
17.	Arithmetic accuracy.	PASSED
18.	Design Logic.	PASSED
19.	Cross-Function race Conditions	PASSED
20.	Save Upon solidity contract Implementation and Usage.	PASSED
21.	Fallback Function Security	PASSED



**AUDIT RESULT**

**PASSED**

SMART CONTRACT AUDIT OF ZUDO FINANCE

Identifier	Definition	Severity
CEN-02	Initial asset distribution	Minor 

All of the initially minted assets are sent to the contract deployer when deploying the contract. This can be an issue as the deployer and/or contract owner can distribute tokens without consulting the community.

}

```
function mint(address _to, uint256 _amount) public onlyOwner {  
    require(  
        _amount + totalSupply() <= MAX_TOTAL_SUPPLY,  
        "ZudoToken: Max total supply exceeded"  
    );  
    _mint(_to, _amount);  
}
```

## RECOMMENDATION

Project stakeholders should be consulted during the initial asset distribution process.



## RECOMMENDATION

**Deployer and/or contract owner private keys are secured carefully.**

**Please refer to PAGE-09 CENTRALIZED PRIVILEGES for a detailed understanding.**

## ALLEVIATION

**The ZUDO FINANCE project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behavior in the project**





Identifier	Definition	Severity
COD-10	Third Party Dependencies	Minor 

Smart contract is interacting with third party protocols e.g., Pancakeswap router, cashier contract, protections contract. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

## RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.



## CERTIFICATE BY VITAL BLOCK SECURITY



## DISCLAIMERS

Vital Block provides the easy-to-understand audit of Solidity, Move and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

## CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

## NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way



to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

**FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.**

### **TECHNICAL DISCLAIMER**

**ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.**

**WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.**

### **TIMELINESS OF CONTENT**

**The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.**



## **LINKS TO OTHER WEBSITES**

**This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.**



## ABOUT VITAL BLOCK

**Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.**

**Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.**

**Website:** <https://Vitalblock.org>

**Email:** [info@vitalblock.org](mailto:info@vitalblock.org)

**GitHub:** <https://github.com/vital-block>

**Telegram (Engineering):** [https://t.me/vital\\_block](https://t.me/vital_block)

**Telegram (Onboarding):** [https://t.me/vitalblock\\_cmo](https://t.me/vitalblock_cmo)





**vital-block**



**info@vitalblock.org**



**www.Vitalblock.org**



Vital Block Dedicated to securing Public and Private Blockchain Ecosystem