SAFUAUDIT
SMART CONTRACT AUDITING

ZENMOON
SMART CONTRACT AUDIT

March 21, 2022

INTRODUCTION

Client	ZenMoon (Zen)
Language	Solidity
Contract address	0x05e81b787d48aEb677463fD909a954104A7c1A40
Decimals	18
Supply	100,000,000,000
Platform	Binance Smart Chain
Compiler	v0.7.6+commit.7338295f
Optimization	Yes, with 200 runs
Website	https://zenmoon.xyz/
Telegram	https://t.me/ZenMoonOfficial
Twitter	https://twitter.com/ZenMoonOfficial

Description

ZenMoon is a hyper-deflationary, rewards token, with a revolutionary, first of its kind, deflationary tax system, hosted on the Binance Smart Chain. There are minimal barriers to entry/exit for new and patient investors, whilst maximal barriers to exit for impatient investors who reward our holders with up to massive 25% BUSD reflections!

TABLE OF CONTENTS

Approach

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

,

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- Remix IDE
- MythX, Mythril
- SWC Registry
- Open Zeppelin Code Analyzer
- Solidity Code Complier

RISK CLASSIFICATION

CRITICAL

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

MEDIUM

Issues on this level could potentially bring problems and should eventually be fixed.

MINOR

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

INFORMATIONAL

Information level is to offer suggestions for improvement of efficacity or security for features with a risk free factor.

ABSTRACT

Fees	Ownership	Mint function
Buy Fees: 10% Sell Fees: variable depending on the holding time	Owned	No mint function found

Owner can set fees	Max Tx amount	Pause
Owner can set fees up to 30%	Owner can set max Tx amount, but not to 0	Owner can't pause trading

Vulnerabilities Test

SWC ID	Description	
SWC-100	Function Default Visibility	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	Minor
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Minor
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed

SWC-116	Block values as a proxy for time	**Passed**
SWC-117	Signature Malleability	**Passed**
SWC-118	Incorrect Constructor Name	**Passed**
SWC-119	Shadowing State Variables	**Passed**
SWC-120	Weak Sources of Randomness from Chain Attributes	**Minor**
SWC-121	Missing Protection against Signature Replay Attacks	**Passed**
SWC-122	Lack of Proper Signature Verification	**Passed**
SWC-123	Requirement Violation	**Passed**
SWC-124	Write to Arbitrary Storage Location	**Passed**
SWC-125	Incorrect Inheritance Order	**Passed**
SWC-126	Insufficient Gas Griefing	**Passed**
SWC-127	Arbitrary Jump with Function Type Variable	**Passed**
SWC-128	DoS With Block Gas Limit	**Passed**
SWC-129	Typographical Error	**Passed**
SWC-130	Right-To-Left-Override control character (U+202E)	**Passed**
SWC-131	Presence of unused variables	**Passed**
SWC-132	Unexpected Ether balance	**Passed**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	**Passed**
SWC-134	Message call with the hardcoded gas amount	**Passed**
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Passed**
SWC-136	Unencrypted Private Data On-Chain	**Passed**

MANUAL ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	Result
Transfer	Yes	**Passed**
Total Supply	Yes	**Passed**
Buy Back	Yes	**N/A**
Burn	Yes	**Passed**
Mint	Yes	**N/A**
Rebase	Yes	**N/A**
Pause	Yes	**N/A**
Blacklist	Yes	**Passed**
Lock	Yes	**N/A**
Max Transaction	Yes	**Passed**
Transfer Ownership	Yes	**Passed**
Renounce Ownership	Yes	**N/A**

MANUAL AUDIT

CONTRACT INSPECTION

SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
IBEP20	Interface			
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
Auth	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	authorize	Public ❗	🛑	onlyOwner
└	unauthorize	Public ❗	🛑	onlyOwner
└	isOwner	Public ❗		NO❗
└	isAuthorized	Public ❗		NO❗
└	transferOwnership	Public ❗	🛑	onlyOwner
IDEXFactory	Interface			
└	createPair	External ❗	🛑	NO❗
IDEXRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗

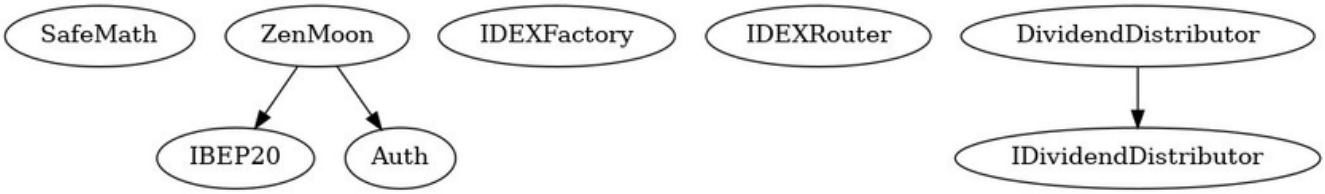
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
IDividendDistributor	Interface			
└	setDistributionCriteria	External ❗	🛑	NO❗
└	setShare	External ❗	🛑	NO❗
└	deposit	External ❗	💵	NO❗
└	process	External ❗	🛑	NO❗
DividendDistributor	Implementation	IDividendDistributor		
└	<Constructor>	Public ❗	🛑	NO❗
└	setDistributionCriteria	External ❗	🛑	onlyToken
└	setShare	External ❗	🛑	onlyToken
└	deposit	External ❗	💵	onlyToken
└	process	External ❗	🛑	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🛑	
└	claimDividend	External ❗	🛑	NO❗
└	getUnpaidEarnings	Public ❗		NO❗
└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🛑	
└	removeShareholder	Internal 🔒	🛑	
ZenMoon	Implementation	IBEP20, Auth		
└	<Constructor>	Public ❗	🛑	Auth
└	<Receive Ether>	External ❗	💵	NO❗
└	totalSupply	External ❗		NO❗
└	decimals	External ❗		NO❗
└	symbol	External ❗		NO❗
└	name	External ❗		NO❗
└	getOwner	External ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	approveMax	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	_transferFrom	Internal 🔒	🛑	

└	_basicTransfer	Internal 🔒	🛑	
└	checkTxLimit	Internal 🔒		
└	shouldTakeFee	Internal 🔒		
└	takeFee	Internal 🔒	🛑	
└	shouldSwapBack	Internal 🔒		
└	swapBack	Internal 🔒	🛑	swapping
└	liquify	Internal 🔒	🛑	
└	buyBack	External ❗	🛑	authorized
└	buyTokens	Internal 🔒	🛑	swapping
└	launched	Internal 🔒		
└	launch	Public ❗	🛑	authorized
└	setTxLimit	External ❗	🛑	authorized
└	setBlacklisted	External ❗	🛑	authorized
└	setIsDividendExempt	External ❗	🛑	authorized
└	startTrading	External ❗	🛑	authorized
└	burnTokens	External ❗	🛑	authorized
└	TransferBNBsOutfromContract	External ❗	🛑	authorized
└	setIsFeeExempt	External ❗	🛑	authorized
└	setIsTxLimitExempt	External ❗	🛑	authorized
└	setFeeReceivers	External ❗	🛑	authorized
└	setSwapBackSettings	External ❗	🛑	authorized
└	setTargetLiquidity	External ❗	🛑	authorized
└	setDistributionCriteria	External ❗	🛑	authorized
└	setDistributorSettings	External ❗	🛑	authorized
└	getCirculatingSupply	Public ❗		NO❗
└	getLiquidityBacking	Public ❗		NO❗
└	isOverLiquified	Public ❗		NO❗
└	setMinTokenAmount	External ❗	🛑	authorized
└	setAddressMaxBuyBlock	Internal 🔒	🛑	
└	getSpenderMaxBuyBlock	Internal 🔒	🛑	
└	getSpenderBuyBlock	Public ❗		NO❗
└	updateTotalTax	External ❗	🛑	authorized
└	updateTaxBlockThreshhold	External ❗	🛑	authorized
└	updateRefTax	External ❗	🛑	authorized
└	updateDevTax	External ❗	🛑	authorized
└	updateMarkTax	External ❗	🛑	authorized
└	updateLPTax	External ❗	🛑	authorized
└	updateMinTokenAmount	External ❗	🛑	authorized
└	getAllTaxBrackets	Public ❗		NO❗

└	getAllTaxBlockThreshholds	Public ❗	|NO❗
└	getCurrentRunningRefDetails	Public ❗	|NO❗
└	calcSellFee	Internal 🔒	🛑	
└	decrementRefCount	Internal 🔒	🛑	
└	refreshReflectionDistPerc	Internal 🔒	🛑	
└	calcFeePerc	Internal 🔒		
└	getFixedTaxes	Public ❗	|NO❗

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable
🔐❗	Private function
🔒	Internal function
NO	Function has no modifier

INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

Important Snippets

Owner can set buy/sell fees up to 30%

```
function updateTotalTax(uint256 index, uint256 value) external authorized {
    require(value <= 30, "Max Tax 30");
    taxBrackets[index] = value;
}


function updateRefTax(uint256 index, uint256 value) external authorized returns(uint256, uint256) {
    require(value <= 30, "Max Ref Tax 30");
    refPercs[index] = value;
    return(refPercs[index],value);
}
```

Owner can set max Tx limit

```
function setTxLimit(uint256 amount) external authorized {
    require(amount >= _totalSupply / 1000);
    _maxTxAmount = amount;
}
```

Owner can exclude from fees

```
function setIsFeeExempt(address holder, bool exempt) external authorized {
    isFeeExempt[holder] = exempt;
}
```

Owner can blacklist certain wallets

```
 function setBlacklisted(address account, bool value) external authorized {
    blackListed[account]= value;
}
```

GOOD PRACTICES

- The owner cannot mint new tokens
- The owner cannot stop or pause the smart contract
- The owner cannot set max TX below 0.1% of Total Supply
- The owner cannot set fees over 30%
- The smart contract utilizes "SafeMath" to prevent overflows

```
library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
    function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }
    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return div(a, b, "SafeMath: division by zero");
    }
    function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {
        // Solidity only automatically asserts when dividing by 0
        require(b > 0, errorMessage);
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in which this doesn't hold

        return c;
    }
}
```

WEBSITE

Website	https://zenmoon.xyz/
Domain Registry	https://www.hostinger.com/
Domain Expiry Date	2023-01-27
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Minor
Robots.txt	Informative
Sitemap Test	Informative
SEO Friendly URL	Informative
Responsive Test	Passed
JS Error Test	Minor
Console Errors Test	Minor
Site Loading Speed Test	2.19 seconds - Passed
HTTP2 Test	Passed
Safe Browsing Test	Passed

DISCLAIMER

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

Accuracy of Information

SafuAudit will strive to ensure accuracy of information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only — we recommend proceeding with several independent audits Be aware that smart contracts deployed on a blockchain aren't secured enough against external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, SafuAudit does not guarantee the explicit security of the audited smart contract. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

AUDIT RESULTS

CRITICAL

No critical severity issues have been found.

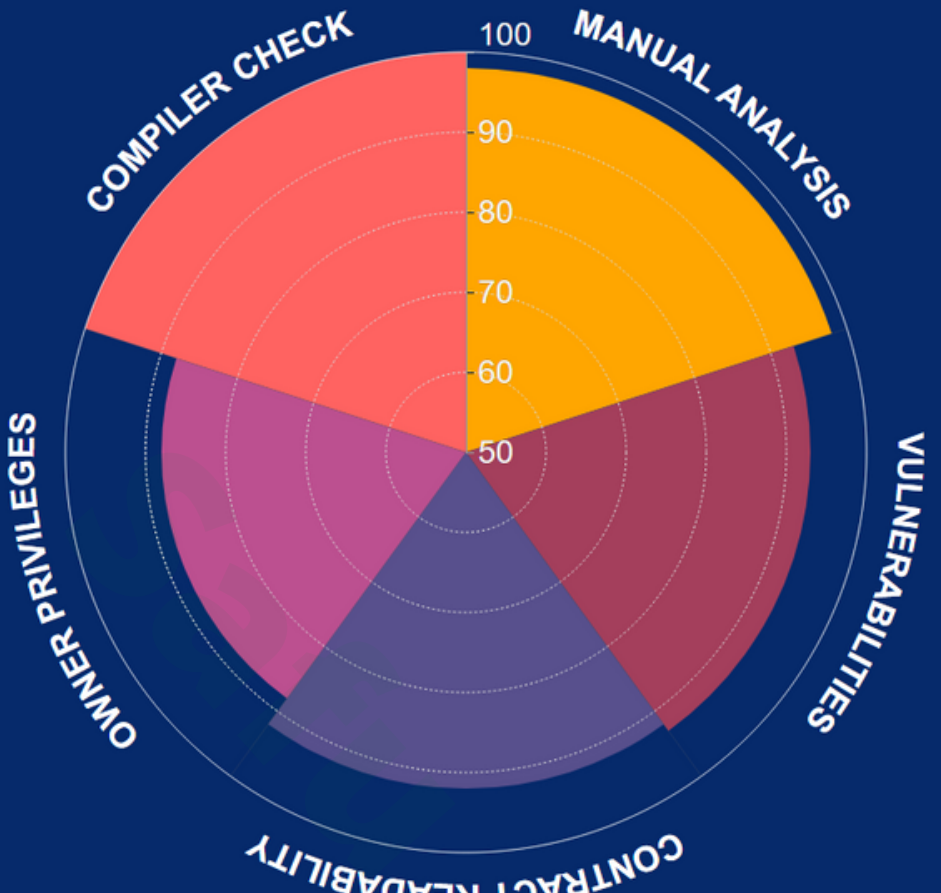
MEDIUM

No medium severity issues have been found.

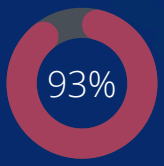
MINOR

- A floating pragma is set. It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.
- State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "_token, BUSD, WBNB, router, shareholders, shareholderIndexes , shareholderClaims, currentIndex, initialized, DEAD, ZERO, DEAD_NON_CHECKSUM, _totalSupply, _balances, _allowances, isFeeExempt, isTxLimitExempt, isDividendExempt, blackListed, _maxBuyBlock, totalRefAmount, _refPercWallet, feeDenominator, taxBrackets, taxBlockThreshholds, refPercs, devPerc, marketingPerc, lpPerc, targetLiquidity, targetLiquidityDenominator, minTokenAmount, start, distributor, distributorGas, inSwap" is internal.
- Potential use of "block.number" as source of randonmness. The environment variable "block.number" looks like it might be used as a source of randomness. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness.

SCORE

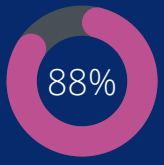


Manual Analysis



Vulnerabilities

Contract Readability



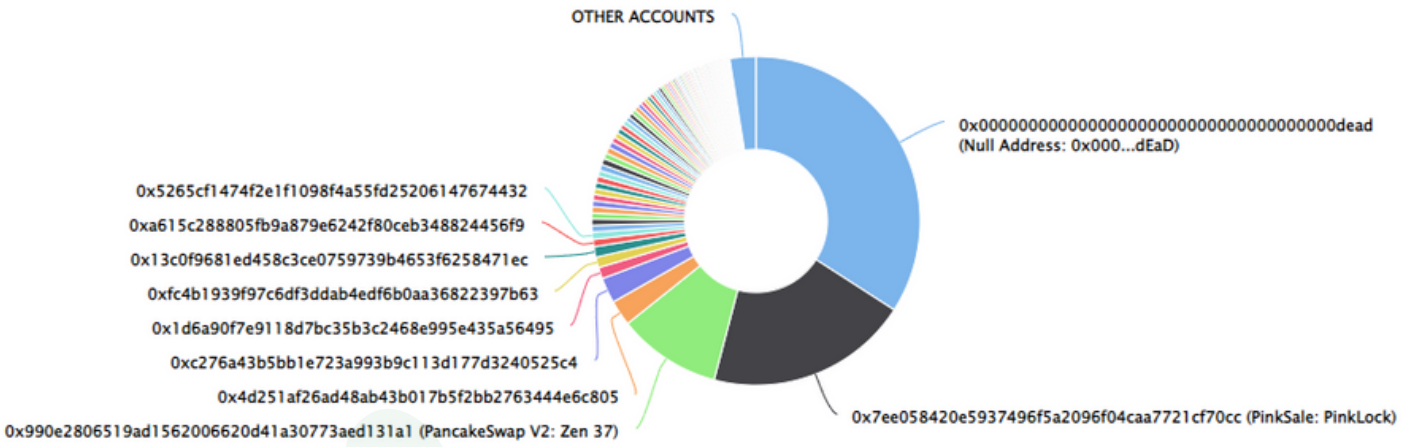
Owner Privileges

Compiler Check

Final Score:94.2

SUMMARY

Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	34,105,000,000	34.1050%
2	PinkSale: PinkLock	20,000,000,000	20.0000%
3	PancakeSwap V2: Zen 37	10,182,178,573.248073727280439489	10.1822%
4	0x4d251af26ad48ab43b017b5f2bb2763444e6c805	2,500,000,000	2.5000%
5	0xc276a43b5bb1e723a993b9c113d177d3240525c4	2,500,000,000	2.5000%
6	0x1d6a90f7e9118d7bc35b3c2468e995e435a56495	1,140,000,000	1.1400%
7	0xfc4b1939f97c6df3ddab4edf6b0aa36822397b63	1,038,028,802.542988321382505275	1.0380%
8	0x13c0f9681ed458c3ce0759739b4653f6258471ec	1,030,288,572.916588006454673472	1.0303%
9	0xa615c288805fb9a879e6242f80ceb348824456f9	719,699,498.791238629996506343	0.7197%
10	0x5265cf1474f2e1f1098f4a55fd25206147674432	716,911,098.257374428811365948	0.7169%

Conclusion

Project ZenMoon does not contain any severe or medium issues.

SafuAudit has tested the security based on manual and automated tests.

Please note that we don't offer any warranties for business model.

SafuAudit.com