SAFUAUDIT
SMART CONTRACT AUDITING

JAGUAR
SMART CONTRACT AUDIT

April 2, 2022

INTRODUCTION

Client	Jaguar (JAGUAR)
Language	Solidity
Contract address	0x7fc402bef360Add6d73371dE181780d3eAd3bA66
Decimals	5
Supply	50,000,000
Platform	Binance Smart Chain
Compiler	v0.7.4+commit.3f05b770
Optimization	Yes, with 200 runs
Website	https://jaguardao.finance/
Telegram	https://t.me/JaguarDao
Twitter	https://twitter.com/jaguardaofin

Description

$JAGUAR is the native token which interest rebase rewards are paid. Every token holder automatically receives 0.03456% interest every 15 minutes just for holding $JAGUAR tokens in their own wallet!

TABLE OF CONTENTS

01 INTRODUCTION

Introduction 02

Approach 04

Risk classification 05

02 ABSTRACT

Abstract 06

03 VULNERABILITIES TEST

Vulnerabilities Test 07

04 MANUAL ANALYSIS

Manual analysis 09

Contract Inspection 10

Inheritance Tree 14

Important Snippets 15

Good Practices 16

05 WEBSITE

Website Audit 17

06 CONCLUSIONS

Disclaimer 18

Audit Results 19

Score 20

Summary 21

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: 12% Sell Fees: 16%	Owned	Rebase Token

Owner can set fees	Max Tx amount	Pause
Owner can't set fees	Owner can't set max Tx amount	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	**Passed**
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	**N/A**
Mint	Yes	**N/A**
Rebase	Yes	**Passed**
Pause	Yes	**N/A**
Blacklist	Yes	**Passed**
Lock	Yes	**N/A**
Max Transaction	Yes	**N/A**
Transfer Ownership	Yes	**Passed**
Renounce Ownership	Yes	**Passed**

CONTRACT INSPECTION

SafeMathInt	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
└	abs	Internal 🔒		
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
IPancakeSwapPair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🛑	NO❗

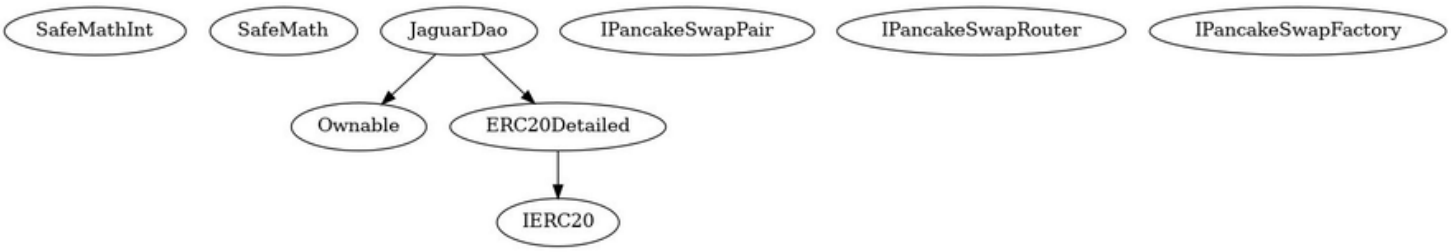
└	MINIMUM_LIQUIDITY	External ❗			NO❗
└	factory	External ❗			NO❗
└	token0	External ❗			NO❗
└	token1	External ❗			NO❗
└	getReserves	External ❗			NO❗
└	price0CumulativeLast	External ❗			NO❗
└	price1CumulativeLast	External ❗			NO❗
└	kLast	External ❗			NO❗
└	mint	External ❗	🛑		NO❗
└	burn	External ❗	🛑		NO❗
└	swap	External ❗	🛑		NO❗
└	skim	External ❗	🛑		NO❗
└	sync	External ❗	🛑		NO❗
└	initialize	External ❗	🛑		NO❗
IPancakeSwapRouter	Interface				
└	factory	External ❗			NO❗
└	WETH	External ❗			NO❗
└	addLiquidity	External ❗	🛑		NO❗
└	addLiquidityETH	External ❗	💵		NO❗
└	removeLiquidity	External ❗	🛑		NO❗
└	removeLiquidityETH	External ❗	🛑		NO❗
└	removeLiquidityWithPermit	External ❗	🛑		NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑		NO❗
└	swapExactTokensForTokens	External ❗	🛑		NO❗
└	swapTokensForExactTokens	External ❗	🛑		NO❗
└	swapExactETHForTokens	External ❗	💵		NO❗
└	swapTokensForExactETH	External ❗	🛑		NO❗
└	swapExactTokensForETH	External ❗	🛑		NO❗
└	swapETHForExactTokens	External ❗	💵		NO❗
└	quote	External ❗			NO❗
└	getAmountOut	External ❗			NO❗
└	getAmountIn	External ❗			NO❗
└	getAmountsOut	External ❗			NO❗
└	getAmountsIn	External ❗			NO❗
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑		NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵		NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑		NO❗

IPancakeSwapFactory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🛑	NO❗
└	setFeeTo	External ❗	🛑	NO❗
└	setFeeToSetter	External ❗	🛑	NO❗
Ownable	Implementation			
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	isOwner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	
ERC20Detailed	Implementation	IERC20		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
JaguarDao	Implementation	ERC20Detailed, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20Detailed Ownable
└	rebase	Internal 🔒	🛑	
└	transfer	External ❗	🛑	validRecipient
└	transferFrom	External ❗	🛑	validRecipient
└	_basicTransfer	Internal 🔒	🛑	
└	_transferFrom	Internal 🔒	🛑	
└	takeFee	Internal 🔒	🛑	
└	addLiquidity	Internal 🔒	🛑	swapping
└	swapBack	Internal 🔒	🛑	swapping
└	withdrawAllToTreasury	External ❗	🛑	swapping onlyOwner
└	shouldTakeFee	Internal 🔒		
└	shouldRebase	Internal 🔒		
└	shouldAddLiquidity	Internal 🔒		
└	shouldSwapBack	Internal 🔒		
└	setAutoRebase	External ❗	🛑	onlyOwner

└	setAutoAddLiquidity	External ❗	🛑	onlyOwner
└	allowance	External ❗		NO❗
└	decreaseAllowance	External ❗	🛑	NO❗
└	increaseAllowance	External ❗	🛑	NO❗
└	approve	External ❗	🛑	NO❗
└	checkFeeExempt	External ❗		NO❗
└	getCirculatingSupply	Public ❗		NO❗
└	isNotInSwap	External ❗		NO❗
└	manualSync	External ❗	🛑	NO❗
└	setFeeReceivers	External ❗	🛑	onlyOwner
└	getLiquidityBacking	Public ❗		NO❗
└	setWhitelist	External ❗	🛑	onlyOwner
└	setBotBlacklist	External ❗	🛑	onlyOwner
└	setPairAddress	Public ❗	🛑	onlyOwner
└	setLP	External ❗	🛑	onlyOwner
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	isContract	Internal 🔒		
└	<Receive Ether>	External ❗	💵	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

Auto rebase

```
function setAutoRebase(bool _flag) external onlyOwner {
    if (_flag) {
        _autoRebase = _flag;
        _lastRebasedTime = block.timestamp;
    } else {
        _autoRebase = _flag;
    }
}
```

Blacklisted contracts are not permitted to transfer their tokens

```
function setBotBlacklist(address _botAddress, bool _flag) external onlyOwner {
    require(isContract(_botAddress), "only contract address, not allowed exteranlly owned account");
    blacklist[_botAddress] = _flag;
}
```

Transfer tokens left in contract to revenue address

```
function withdrawAllToTreasury() external swapping onlyOwner {

    uint256 amountToSwap = _gonBalances[address(this)].div(_gonsPerFragment);
    require( amountToSwap > 0,"There is no Jaguar token deposited in token contract");
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = router.WETH();
    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        amountToSwap,
        0,
        path,
        treasuryReceiver,
        block.timestamp
    );
}
```

GOOD PRACTICES

- The owner cannot stop or pause the smart contract
- The owner cannot set the fees
- The owner cannot set max Tx
- The smart contract utilizes "SafeMath" to prevent overflows

```
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,
        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,
        string memory errorMessage
    ) internal pure returns (uint256) {
        require(b > 0, errorMessage);
        uint256 c = a / b;
        return c;
    }
   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0);
        return a % b;
    }
```

WEBSITE

Website	https://jaguardao.finance/
Domain Registry	http://www.hostinger.com
Domain Expiry Date	2023-03-08
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Informative
Sitemap Test	Informative
SEO Friendly URL	Informative
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	2.17 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.

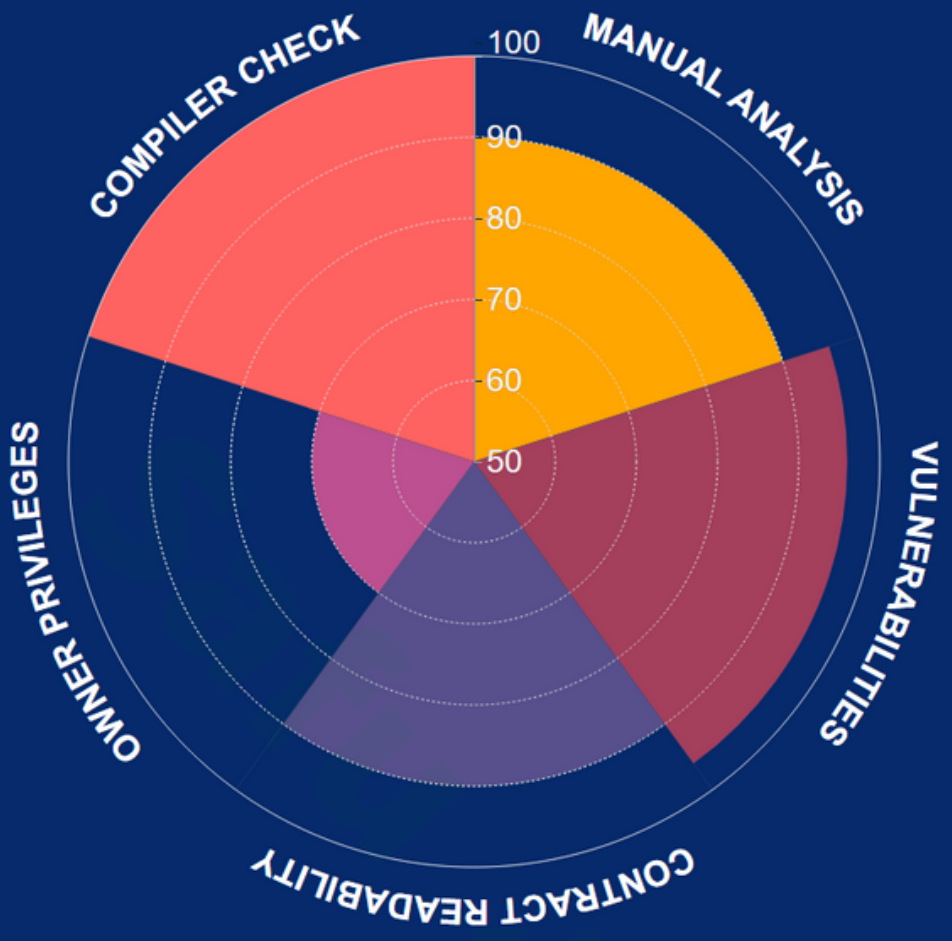
MINOR

- 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. This is especially important if you rely on bytecode-level verification of the code.
- State variable visibility is not set for: _isFeeExempt, inSwap, DEAD, ZERO

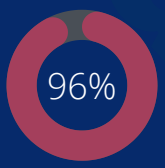
INFORMATIONAL

- The owner can toggle liquidity add and rebase functionality at any time.
- The owner can swap all $JAGUAR in the contract for BNB and withdraw it from the contract to the "treasuryReceiver" address at any time.

SAFUSCORE

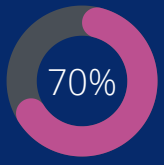


Manual Analysis



Vulnerabilities

Contract Readability



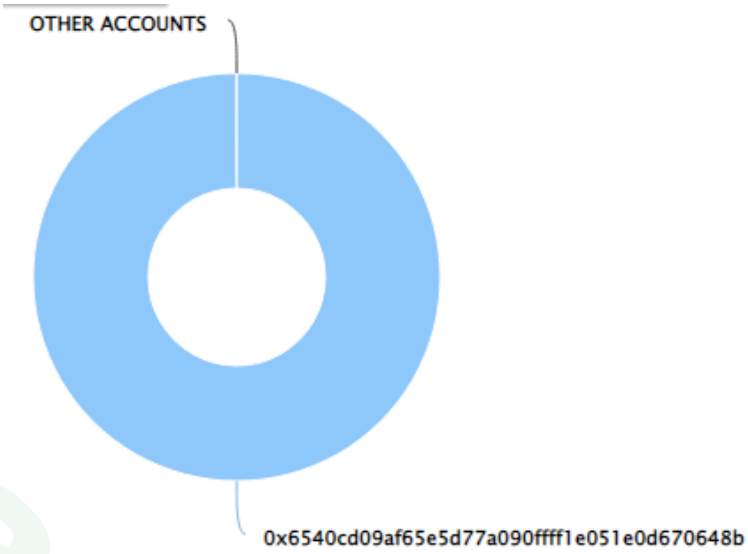
Owner Privileges

Compiler Check

Final Score:89.2

SUMMARY

Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	0x6540cd09af65e5d77a090ffff1e051e0d670648b	50,000,000	100.0000%

Conclusion

Project Jaguar (JAGUAR) does not contain any severe issues. This is a rebase token: the owner can toggle liquidity adds and rebases from occurring at any time.

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