SAFUAUDIT
SMART CONTRACT AUDITING

CAHU
SMART CONTRACT AUDIT

March 23, 2022

INTRODUCTION

Client	Cahu.com (CAHU)
Language	Solidity
Contract Address	0x7A1CBaFb22004156728c149b002B09A02Db72C9E
Decimals	18
Supply	1 CAHU
Platform	Binance Smart Chain
Compiler	v0.6.12+commit.27d51765
Optimization	Yes, with 200 runs
Website	https://www.cahu.com/
Telegram	https://twitter.com/Cahu
Twitter	https://t.me/Cahuuu

Description

$CAHU is a new algorithmic #BNB stablecoin that fuses CEX and DEX liquidity into a community-owned token engine of Cahu Reward ($CAR) and Cahu Bond ($CAB)!

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: 0% Sell Fees: 0% *at audit time	Owned	Yes

Owner can set fees	Max Tx amount	Pause
Owner can't set fees over 9.99%	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SWC-102	Outdated Compiler Version	Passed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	Passed
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	**Passed**
Mint	Yes	**Passed**
Rebase	Yes	**N/A**
Pause	Yes	**N/A**
Blacklist	Yes	**N/A**
Lock	Yes	**N/A**
Max Transaction	Yes	**N/A**
Transfer Ownership	Yes	**Passed**
Renounce Ownership	Yes	**Passed**

CONTRACT INSPECTION

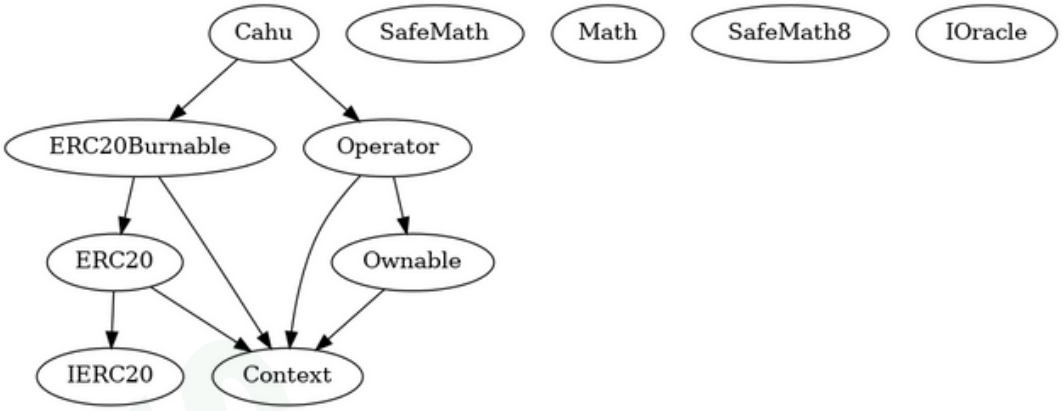
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
└	tryAdd	Internal 🔒		
└	trySub	Internal 🔒		
└	tryMul	Internal 🔒		
└	tryDiv	Internal 🔒		
└	tryMod	Internal 🔒		
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	sub	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
ERC20	Implementation	Context, IERC20		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗

└	transferFrom	Public ❗	🛑	NO❗
└	increaseAllowance	Public ❗	🛑	NO❗
└	decreaseAllowance	Public ❗	🛑	NO❗
└	_transfer	Internal 🔒	🛑	
└	_mint	Internal 🔒	🛑	
└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_setupDecimals	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
ERC20Burnable	Implementation	Context, ERC20		
└	burn	Public ❗	🛑	NO❗
└	burnFrom	Public ❗	🛑	NO❗
Math	Library			
└	max	Internal 🔒		
└	min	Internal 🔒		
└	average	Internal 🔒		
SafeMath8	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
Operator	Implementation	Context, Ownable		
└	<Constructor>	Internal 🔒	🛑	
└	operator	Public ❗		NO❗
└	isOperator	Public ❗		NO❗

└	transferOperator	Public ❗	🛑	onlyOwner
└	_transferOperator	Internal 🔒	🛑	
IOracle	Interface			
└	update	External ❗	🛑	NO❗
└	consult	External ❗		NO❗
└	twap	External ❗		NO❗
Cahu	Implementation	ERC20Burnable, Operator		
└	<Constructor>	Public ❗	🛑	ERC20
└	getTaxTiersTwapsCount	Public ❗		NO❗
└	getTaxTiersRatesCount	Public ❗		NO❗
└	isAddressExcluded	Public ❗		NO❗
└	setTaxTiersTwap	Public ❗	🛑	onlyTaxOffice
└	setTaxTiersRate	Public ❗	🛑	onlyTaxOffice
└	setBurnThreshold	Public ❗	🛑	onlyTaxOffice
└	_getCahuPrice	Internal 🔒		
└	_updateTaxRate	Internal 🔒	🛑	
└	enableAutoCalculateTax	Public ❗	🛑	onlyTaxOffice
└	disableAutoCalculateTax	Public ❗	🛑	onlyTaxOffice
└	setCahuOracle	Public ❗	🛑	onlyOperatorOrTaxOffice
└	setTaxOffice	Public ❗	🛑	onlyOperatorOrTaxOffice
└	setTaxCollectorAddress	Public ❗	🛑	onlyTaxOffice
└	setTaxRate	Public ❗	🛑	onlyTaxOffice
└	excludeAddress	Public ❗	🛑	onlyOperatorOrTaxOffice
└	includeAddress	Public ❗	🛑	onlyOperatorOrTaxOffice
└	mint	Public ❗	🛑	onlyOperator
└	burn	Public ❗	🛑	NO❗
└	burnFrom	Public ❗	🛑	onlyOperator
└	transferFrom	Public ❗	🛑	NO❗
└	_transferWithTax	Internal 🔒	🛑	
└	distributeReward	External ❗	🛑	onlyOperator
└	governanceRecoverUnsupported	External ❗	🛑	onlyOperator

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

Notice: This is a stablecoin type of token. Functionalities like minting or burning are necessary, but need to be carefully used.

Burn functions

```
function burn(uint256 amount) public override {
    super.burn(amount);
}

function burnFrom(address account, uint256 amount) public override onlyOperator {
    super.burnFrom(account, amount);
}
```

Mint function

```
function mint(address recipient_, uint256 amount_) public onlyOperator returns (bool) {
    uint256 balanceBefore = balanceOf(recipient_);
    _mint(recipient_, amount_);
    uint256 balanceAfter = balanceOf(recipient_);

    return balanceAfter > balanceBefore;
}
```

Setting tax rates lower than 10%

TaxOffice is 0 address - this cannot be modified anymore

```
function setTaxRate(uint256 _taxRate) public onlyTaxOffice {
    require(!autoCalculateTax, "auto calculate tax cannot be enabled");
    require(_taxRate < 1000, "tax equal or bigger to 10%");
    taxRate = _taxRate;
}
```

GOOD PRACTICES

- The owner cannot set max Tx amount
- The owner cannot stop or pause the smart contract
- The owner cannot set fees over 9.99%
- 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) {
        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
        // benefit is lost if 'b' is also tested.
        // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522
        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) {
        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;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return mod(a, b, "SafeMath: modulo by zero");
    }

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

WEBSITE

Website	https://www.cahu.com/
Domain Registry	http://www.uniregistry.com
Domain Expiry Date	2022-11-15
Response Code	200
SSL Checker and HTTPS Test	Passed
Deprecated HTML tags	Passed
Robots.txt	Informational
Sitemap Test	Passed
SEO Friendly URL	Passed
Responsive Test	Passed
JS Error Test	Passed
Console Errors Test	Passed
Site Loading Speed Test	1.76 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

- Owner privileges: Mint function is accessible to owner and can generate new tokens.

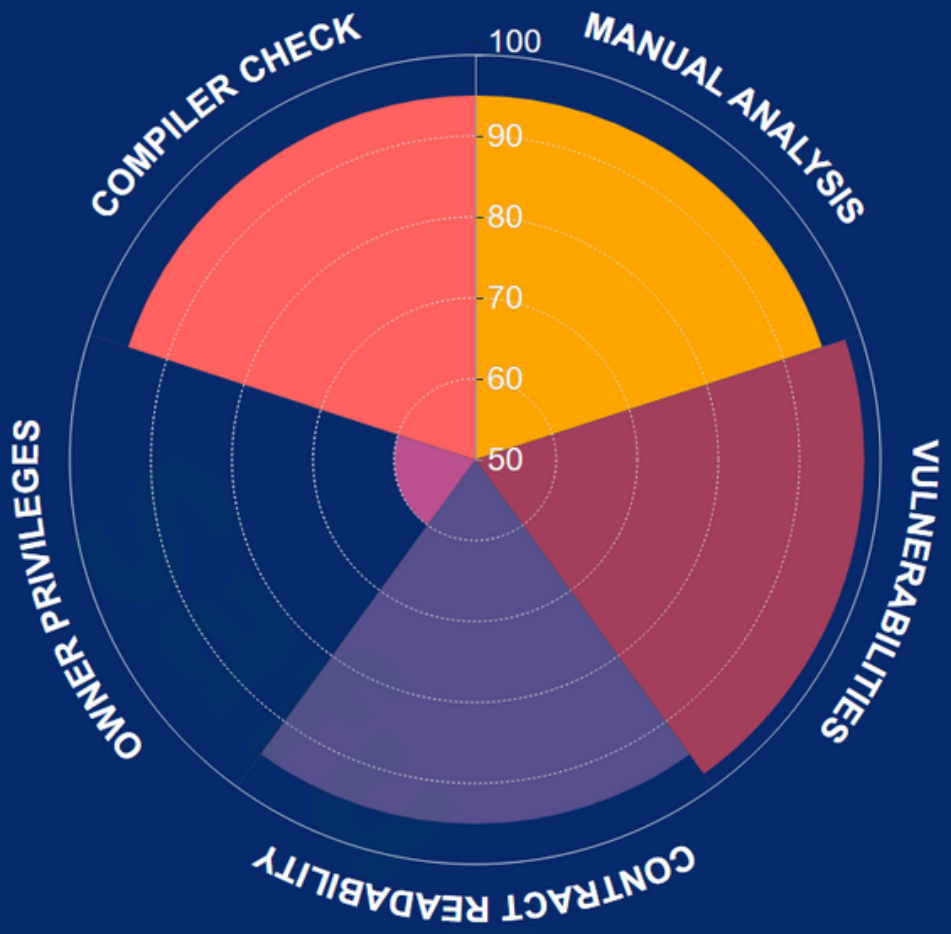
MINOR

- 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.

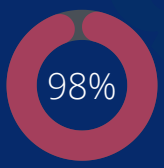
INFORMATIONAL

The standard audit model does not offer suggestions and consulting for improvements of efficacity.

SCORE

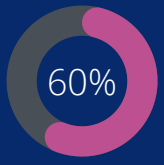


Manual Analysis



Vulnerabilities

Contract Readability



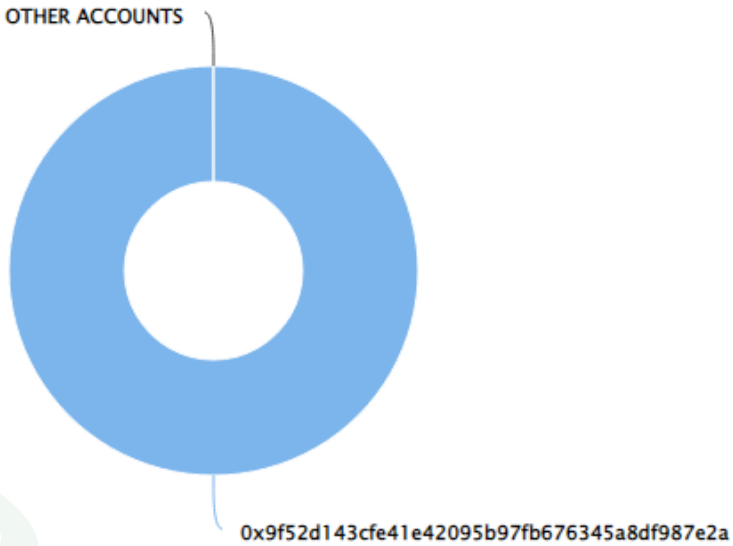
Owner Privileges

Compiler Check

Final Score:88.6

SUMMARY

Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	0x9f52d143cfe41e42095b97fb676345a8df987e2a	1	100.0000%

Conclusion

Project Cahu does not contain any severe issues. Owner can can mint new tokens. According to the owner 8000 CAHU will be minted in the presale, 3000 on the Genesis, after the sale.

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