SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

CHAD CHAIN

ChadChain CHAD 2.0 [$C2.0]

BEP 20

0xc02DaF902dc537cE1cf62095828aD37426654873

Table of Contents

Global Disclaimer

This document serves as a disclaimer for the crypto smart contract audit conducted by Skeleton Ecosystem. The purpose of the audit was to review the codebase of the smart contracts for potential vulnerabilities and issues. It is important to note the following:

Limited Scope: The audit is based on the code and information available up to the audit completion date. It does not cover external factors, system interactions, or changes made after the audit. The audit itself can not guarantee 100% safaty and can not detect common scam methods like farming and developer sell-out.

No Guarantee of Security: While we have taken reasonable steps to identify vulnerabilities, it is impossible to guarantee the complete absence of security risks or issues. The audit report provides an assessment of the contract’s security as of the audit date.

Continued Development: Smart contracts and blockchain technology are evolving fields. Updates, forks, or changes to the contract post-audit may introduce new risks that were not present during the audit.

Third-party Code: If the smart contract relies on third-party libraries or code, those components were not thoroughly audited unless explicitly stated. Security of these dependencies is the responsibility of their respective developers.

Non-Exhaustive Testing: The audit involved automated analysis, manual review, and testing under controlled conditions. It is possible that certain vulnerabilities or issues may not have been identified.

Risk Evaluation: The audit report includes a risk assessment for identified vulnerabilities. It is recommended that the development team carefully reviews and addresses these risks to mitigate potential exploits.

Not Financial Advice: This audit report is not intended as financial or investment advice. Decisions regarding the use, deployment, or investment in the smart contract should be made based on a comprehensive assessment of the associated risks.

By accessing and using this audit report, you acknowledge and agree to the limitations outlined above. Skeleton Ecosystem and its auditors shall not be held liable for any direct or indirect damages resulting from the use of the audit report or the smart contract itself.

Please consult with legal, technical, and financial professionals before making any decisions related to the smart contract.

Overview

Contract Name	CHAD 2.0
Ticker/Simbol	C2.0
Blockchain	Binance Smart Chain BEP20
Contract Address	0xc02DaF902dc537cE1cf62095828aD37426654873
Creator Address	0x74E885903457a64ac318a0c425ec430De6Ac75F6
Current Owner Address	0x74E885903457a64ac318a0c425ec430De6Ac75F6
Contract Explorer	https://bscscan.com/token/0xc02daf902dc537ce1cf62095828ad37426654873
Compiler Version	v0.8.7+commit.e28d00a7
License	None
Optimisation	Yes with 200 Runs
Total Supply	1,000,000 **C2.0**
Decimals	9

Creation/Audit

Contract Deployed	08 July 2023
Audit Created	09 Nov 2023
Audit Update	V 1.0

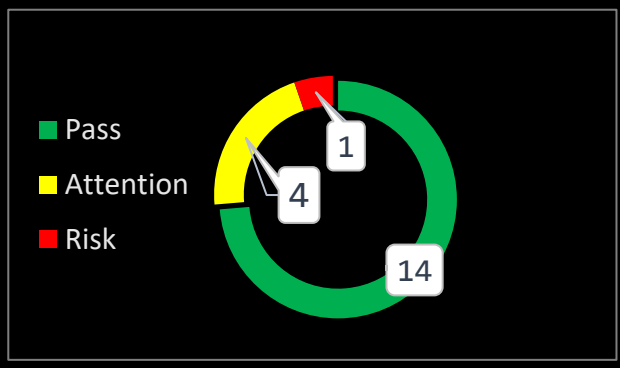
Verified Socials

Website	https://chadchain.org/
Telegram	https://t.me/ChadChainBlockchain
Twitter (X)	http://x.com/ChadChainOfc

CHADCHAIN CHAD 2.0 BEP20

Contract Function Analysis

Pass Attention Item Risky Item



Contract Verified	✅	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x74E885903457a64ac318a0c425ec430De6Ac75F6
Buy Tax	9 %	Shows the taxes for purchase transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Sell Tax	14 %	Shows the taxes for sell transactions. Above 10% may be considered a high tax rate. More than 50% tax rate means may not be tradable. Fee can be set!
Honeypot Analyse	✅	Holder is able to buy and sell. If honeypot: The contract blocks sell transfer from holder wallet. Multiple events may cause honeypot. Trading disabled, extremely high tax
Liqudity Status	✅	LP Lock Status on 09.11.2023: 80.20% Mudra Locker for *204 days*.
Trading Disable Functions	⚠	Trading suspendable function found, contract renounced If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used
Set Fees function	⚠	Fee Setting function found, contract renounced The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded (honeypot risk).
Proxy Contract	✅	Not a proxy contract!
Mint Function	✅	No Mint Function detected 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. Owner can mint new tokens and sell.

Balance Modifier Function	✅	No Balance Modifier function found. If there is a function for this, the contract owner can have the authority to modify the balance of tokens at other addresses. For example revoke the bought tokens from the holders wallet. Common form of scam: You buy the token, but it's disappearing from your wallet.
Blacklist Function	⚠️	Blacklist Setting function, contract renounced If there is a blacklist, some addresses may not be able to trade normally. Example: you buy the token and right after your Wallet getting blacklisted. Like so you will be unable to sell. Honeypot Risk.
Whitelist Function	⚠️	Whitelist Setting function found, Contract renounced If there is a function for this Developer can set zero fee or no max wallet size for adresses (for example team wallets can trade without fee. Can cause farming)
Hidden Owner Analysis	✅	No hidden or multi owner For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.
Retrieve Ownership Function	✅	No functions found which can retrieve ownership of the contract. If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.
Self Destruct Function	✅	No Self Destruct function found. If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.
Specific Tax Changing Function	✅	No Specific Tax Changing Functions found. If it exists, the contract owner may set a very outrageous tax rate for assigned address to block it from trading. Can assign all wallets at once!
Trading Cooldown Function	✅	No Trading Cooldown Function found. 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. Like a temporary honeypot.
Max Transaction and Holding Modify Function	⚠️	Max Transaction and Holding Modify function found, contract renounced If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot
Transaction Limiting Function	✅	No Transaction Limiter Function Found. The number of overall token transactions may be limited (honeypot risk)

Details of Risk - Attention Items

⚠ Set Fee (remedation: renounce ownership)

The contract owner may contain the authority to modify the transaction tax. If the transaction tax is increased to more than 49%, the tokens may not be able to be traded

```
ftrace | funcSig
function setTaxes(uint256 _rfi, uint256 _marketing, uint256 _liquidity, uint256 _donation) public onlyOwner {
    taxes = Taxes(_rfi,_marketing,_liquidity,_donation);
    emit FeesChanged();
}

ftrace | funcSig
function setSellTaxes(uint256 _rfi, uint256 _marketing, uint256 _liquidity, uint256 _donation) public onlyOwner {
    sellTaxes = Taxes(_rfi,_marketing,_liquidity,_donation);
    emit FeesChanged();
}
```

⚠ Whitelist (Set excluded) (remedation: renounce ownership)

If there is a function for this Developer can set zero fee or no max wallet size for adresses (for example team wallets can trade without fee)

```
ftrace | funcSig
function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}
```

⚠ Max Transaction and Holding Modify Function (remedation: renounce ownership)

If there is a function for this, the maximum trading amount or maximum position can be modified.

```
ftrace | funcSig
function updateMaxTxLimit(uint256 maxBuy, uint256 maxSell) external onlyOwner{
    maxBuyLimit = maxBuy * 10**decimals();
    maxSellLimit = maxSell * 10**decimals();
}

ftrace | funcSig
function updateMaxWalletlimit(uint256 amount) external onlyOwner{
    maxWalletLimit = amount * 10**decimals();
}
```

```
uint256 public swapTokensAtAmount = 5_000 * 10**9;
uint256 public maxBuyLimit = 10_000 * 10**9;
uint256 public maxSellLimit = 10_000 * 10**9;
uint256 public maxWalletLimit = 20_000 * 10**9;
```

Trading Suspendable Function (remedation: renounce ownership)

If a suspendable code is included, the token maybe neither be bought or sold (honeypot risk). If contract is renounced this function can't be used

```
ftrace | funcSig
function setTradingStatus(bool state1) external onlyOwner{
    tradingEnabled = state1;
    swapEnabled = state1;
    if(state1 == true && genesis_block == 0) genesis_block = block.number;
}
```

Blacklist Function (remedation: renounce ownership)

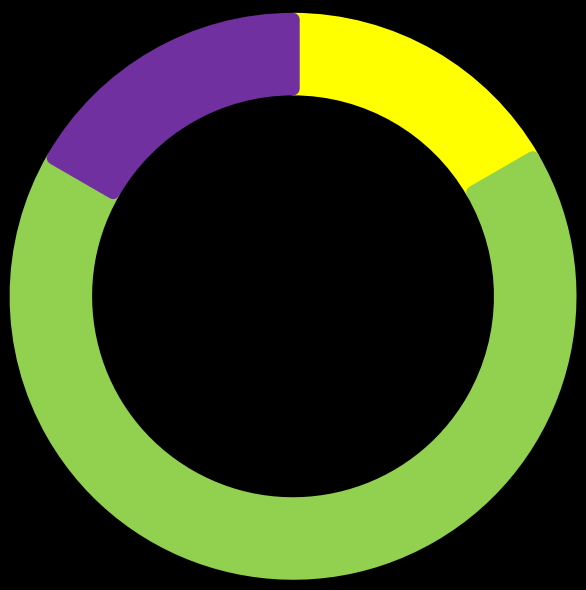
If there is a blacklist, some addresses may not be able to trade normally. Example: you buy the token and right after your Wallet getting blacklisted. Like so you will be unable to sell. Honeypot Risk.

```
ftrace | funcSig
function updateIsBlacklisted(address account1, bool state1) external onlyOwner{
    _isBlacklisted[account1] = state1;
}

ftrace | funcSig
function bulkIsBlacklisted(address[] memory accounts1, bool state1) external onlyOwner{
    for(uint256 i =0; i < accounts1.length; i++){
        _isBlacklisted[accounts1[i]] = state1;

    }
}
```

Contract Security

Total Findings: 6

- **High 0**
- **Medium 1**
- **Low 4**
- **Info 1**



High Severity Issues: High possibility to cause problems, need to be resolved.

Medium Severity Issue: Will likely cause problems, recommended to resolve.

Low Severity Issues: Won’t cause problems, but for improvement purposes could be adjusted.

Informational Severity Issues: Not harmful in any way, information for the developer team.

Contract Security

List of Found Issues

High severity Issues: (0)

Medium severity issues: (1)

- Unchecked Array Lenght

Low severity issues: (4)

- Missing Events
- Long Number Literals
- Floating Pragma
- Outdated Compiler Version

Informational severity issues: (1)

- Public Functions Should be Declared External

Contract Weakness Classisication

THE SMART CONTRACT WEAKNESS CLASSIFICATION REGISTRY (SWC REGISTRY) IS AN IMPLEMENTATION OF THE WEAKNESS CLASSIFICATION SCHEME PROPOSED IN EIP-1470. IT IS LOOSELY ALIGNED TO THE TERMINOLOGIES AND STRUCTURE USED IN THE COMMON WEAKNESS ENUMERATION (CWE) WHILE OVERLAYING A WIDE RANGE OF WEAKNESS VARIANTS THAT ARE SPECIFIC TO SMART CONTRACTS.

ID	Description	AI	Manual	Result
SWC-100	Function Default Visibility	Passed	Passed	Passed
SWC-101	Integer Overflow and Underflow	Passed	Passed	Passed
SWC-102	Outdated Compiler Version	Low	Passed	Passed
SWC-103	Floating Pragma	Low	Passed	Passed
SWC-104	Unchecked Call Return Value	Passed	Passed	Passed
SWC-105	Unprotected Ether Withdrawal	Passed	Passed	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed	Passed	Passed
SWC-107	Reentrancy	Passed	Passed	Passed
SWC-108	State Variable Default Visibility	Passed	Passed	Passed
SWC-109	Uninitialized Storage Pointer	Passed	Passed	Passed
SWC-110	Assert Violation	Passed	Passed	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed	Passed	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed	Passed	Passed
SWC-113	DoS with Failed Call	Passed	Passed	Passed
SWC-114	Transaction Order Dependence	Passed	Passed	Passed
SWC-115	Authorization through tx.origin	Passed	Passed	Passed
SWC-116	Block values as a proxy for time	Passed	Passed	Passed
SWC-117	Signature Malleability	Passed	Passed	Passed
SWC-118	Incorrect Constructor Name	Passed	Passed	Passed
SWC-119	Shadowing State Variables	Passed	Passed	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	Passed	Passed

SWC-121	Missing Protection against Signature Replay Attacks	Passed	Passed	Passed
SWC-122	Lack of Proper Signature Verification	Passed	Passed	Passed
SWC-123	Requirement Violation	Passed	Passed	Passed
SWC-124	Write to Arbitrary Storage Location	Passed	Passed	Passed
SWC-125	Incorrect Inheritance Order	Passed	Passed	Passed
SWC-126	Insufficient Gas Griefing	Passed	Passed	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed	Passed	Passed
SWC-128	DoS With Block Gas Limit	Passed	Passed	Passed
SWC-129	Typographical Error	low	Passed	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed	Passed	Passed
SWC-131	Presence of unused variables	Passed	Passed	Passed
SWC-132	Unexpected Ether balance	Passed	Passed	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed	Passed	Passed
SWC-134	Message call with hardcoded gas amount	Passed	Passed	Passed
SWC-135	Code With No Effects	Passed	Passed	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed	Passed	Passed

Detected High and Medium Severity Vulnerability Description.

⚠ Unchecked Array Lenght (3 Item)

Item: 1	Location:	Line 328	Severity:	Medium
Item: 2	Location:	Line 617	Severity:	Medium
Item: 3	Location:	Line 649	Severity:	Medium

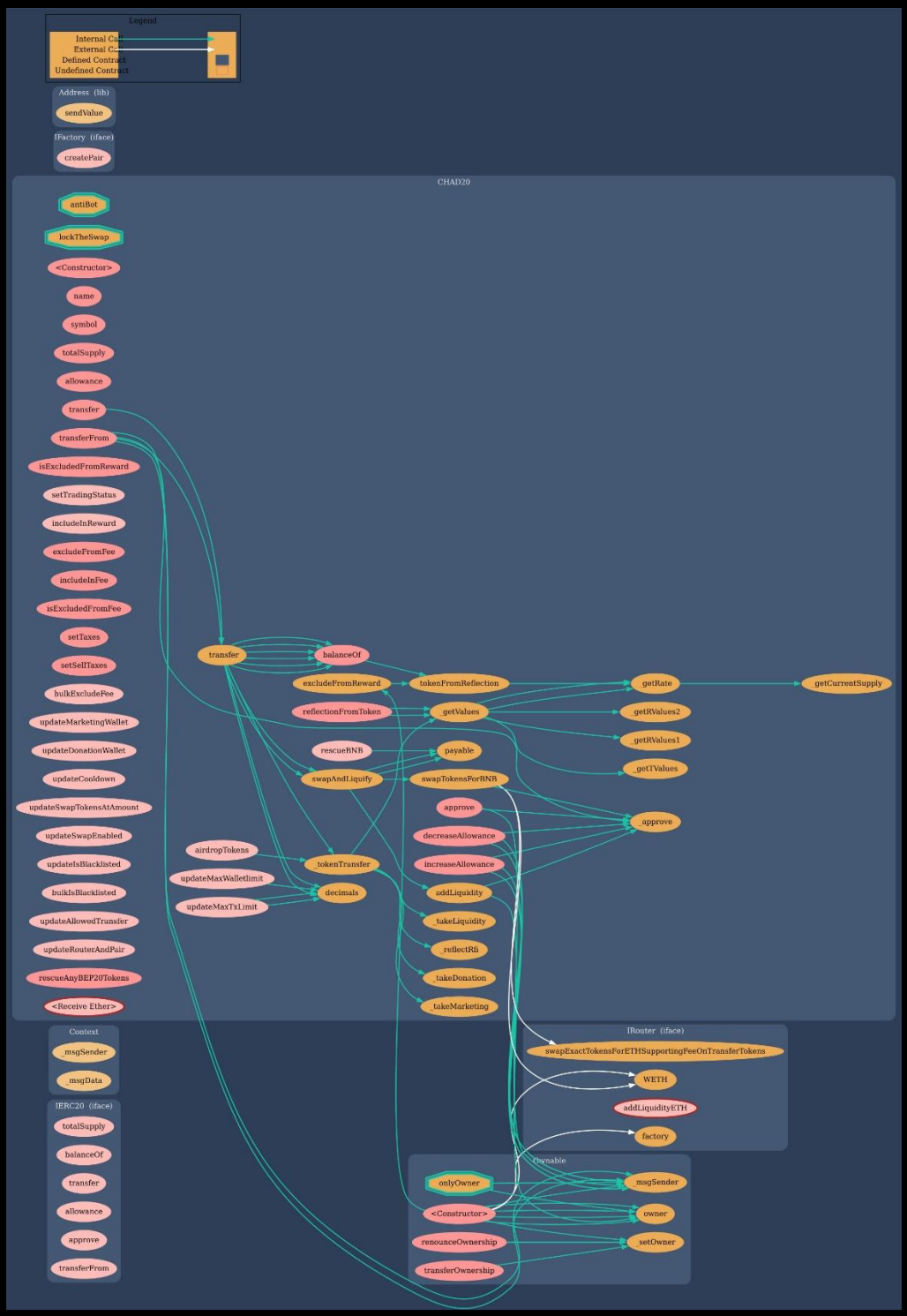
Function	Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If array.length is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed. for (uint256 i = 0; i < array.length ; i++) { costlyFunc(); } This becomes a security issue if an external actor influences array.length. E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.
Remedation	Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

```
327        require(_isExcluded[account], "Account is not excluded");
328        for (uint256 i = 0; i < _excluded.length; i++) {
```

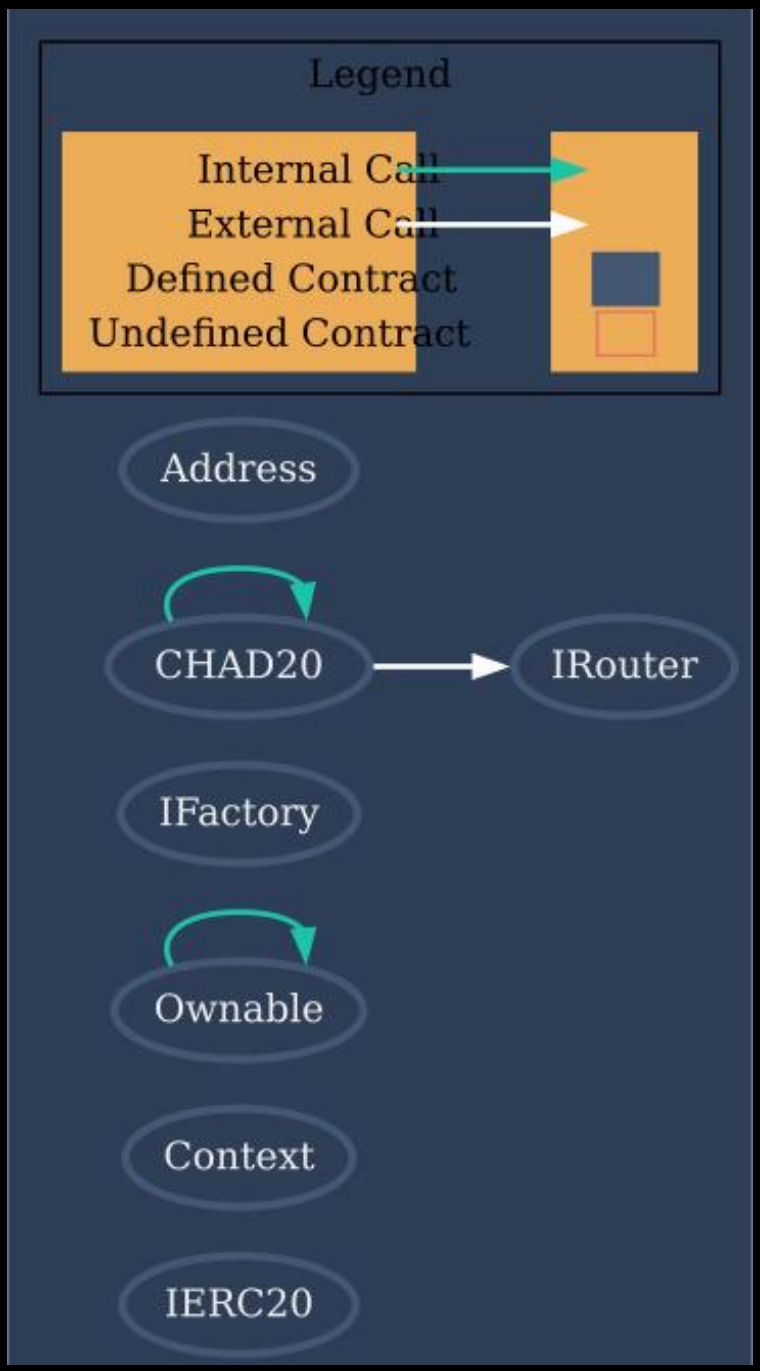
```
616    function bulkExcludeFee(address[] memory accounts, bool state) external onlyOwner{
617        for(uint256 i = 0; i < accounts.length; i++){
```

```
ftrace | funcSig
648    function bulkIsBlacklisted(address[] memory accounts, bool state) external onlyOwner{
649        for(uint256 i =0; i < accounts.length; i++){
```

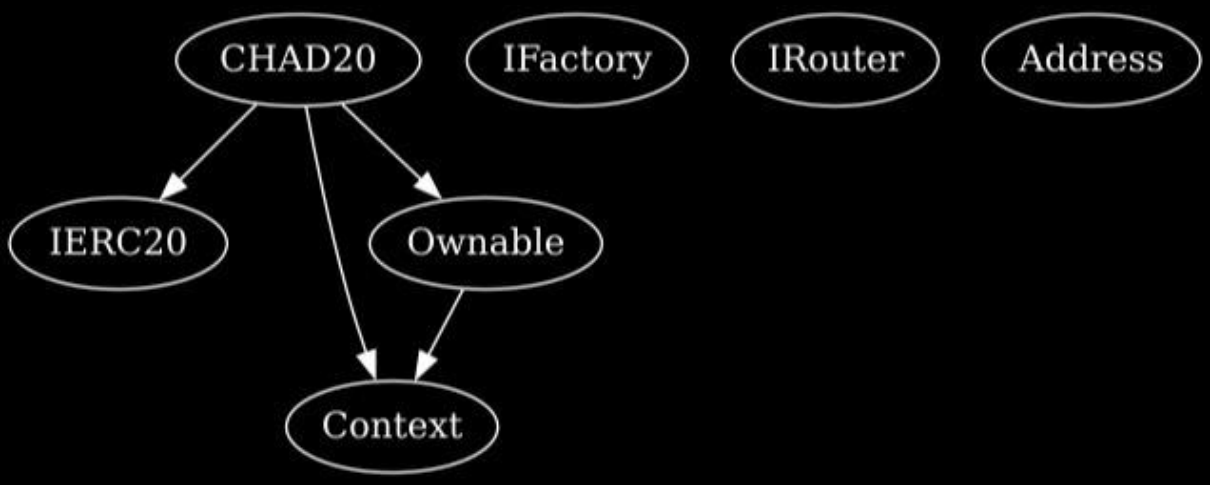
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
L	totalSupply	External !		NO!
L	balanceOf	External !		NO!
L	transfer	External !		NO!
L	allowance	External !		NO!
L	approve	External !		NO!
L	transferFrom	External !		NO!
Context	Implementation			
L	_msgSender	Internal		
L	_msgData	Internal		
Ownable	Implementation	Context		
L		Public !		NO!
L	owner	Public !		NO!
L	renounceOwnership	Public !		onlyOwner
L	transferOwnership	Public !		onlyOwner
L	_setOwner	Private		
IFactory	Interface			
L	createPair	External !		NO!

Contract	Type	Bases		
IRouter	Interface			
L	factory	External !		NO!
L	WETH	External !		NO!
L	addLiquidityETH	External !		NO!
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External !		NO!
Address	Library			
L	sendValue	Internal		
CHAD20	Implementation	Context, IERC20, Ownable		
L		Public !		NO!
L	name	Public !		NO!
L	symbol	Public !		NO!
L	decimals	Public !		NO!
L	totalSupply	Public !		NO!
L	balanceOf	Public !		NO!
L	allowance	Public !		NO!
L	approve	Public !		antiBot
L	transferFrom	Public !		antiBot
L	increaseAllowance	Public !		antiBot
L	decreaseAllowance	Public !		antiBot

Contract	Type	Bases		
L	transfer	Public ❗	🛑	antiBot
L	isExcludedFromReward	Public ❗		NO❗
L	reflectionFromToken	Public ❗		NO❗
L	setTradingStatus	External ❗	🛑	onlyOwner
L	tokenFromReflection	Public ❗		NO❗
L	excludeFromReward	Public ❗	🛑	onlyOwner
L	includeInReward	External ❗	🛑	onlyOwner
L	excludeFromFee	Public ❗	🛑	onlyOwner
L	includeInFee	Public ❗	🛑	onlyOwner
L	isExcludedFromFee	Public ❗		NO❗
L	setTaxes	Public ❗	🛑	onlyOwner
L	setSellTaxes	Public ❗	🛑	onlyOwner
L	_reflectRfi	Private 🔐	🛑	
L	_takeLiquidity	Private 🔐	🛑	
L	_takeMarketing	Private 🔐	🛑	
L	_takeDonation	Private 🔐	🛑	
L	_getValues	Private 🔐		
L	_getTValues	Private 🔐		
L	_getRValues1	Private 🔐		

Contract	Type	Bases		
L	_getRValues2	Private		
L	_getRate	Private		
L	_getCurrentSupply	Private		
L	_approve	Private		
L	_transfer	Private		
L	_tokenTransfer	Private		
L	swapAndLiquify	Private		lockTheSwap
L	addLiquidity	Private		
L	swapTokensForBNB	Private		
L	airdropTokens	External		onlyOwner
L	bulkExcludeFee	External		onlyOwner
L	updateMarketingWallet	External		onlyOwner
L	updateDonationWallet	External		onlyOwner
L	updateCooldown	External		onlyOwner
L	updateSwapTokensAtAmount	External		onlyOwner
L	updateSwapEnabled	External		onlyOwner
L	updateIsBlacklisted	External		onlyOwner
L	bulkIsBlacklisted	External		onlyOwner

Contract	Type	Bases		
L	updateAllowedTransfer	External !	🛑	onlyOwner
L	updateMaxTxLimit	External !	🛑	onlyOwner
L	updateMaxWalletlimit	External !	🛑	onlyOwner
L	updateRouterAndPair	External !	🛑	onlyOwner
L	rescueBNB	External !	🛑	onlyOwner
L	rescueAnyBEP20Tokens	Public !	🛑	onlyOwner
L		External !	💵	NO!

🛑 Function can modify state

💵 Function is payable

Audit Scope

Audit Method.

Our smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. Goal: discover errors, issues and security vulnaribilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review
We are using automated tools to scan functions and weeknesses of the contract. Transfers, integer over-undeflow checks such as all CWE events.

Tools we use:
Visual Studio Code
CWE
SWC
Solidity Scan
SVD

In manual code review our auditor looking at source code and performing line by line examination. This method helps to clarify developer's coding decisions and business logic.

Skeleton Ecosystem

https://skeletonecosystem.com

https://github.com/SkeletonEcosystem/Audits