

SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



LITTLE TETHER
LITE
BEP20

0x12bF34d7418A9025d6B3a54e42b993c43122



Table of Contents

Table of Contents	1
Disclaimer	2
Overview	3
Creation/Audit Date	3
Verified Socials	3
Contract Functions Analysis	4
Contract Safety and Weakness	9
Detected Vulnerability Description	13
Contract Flow Graph	14
Contract Interaction Graph	15
Inheritance Graph	16
Contract Descriptions	17
Audit Scope	24

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% safety 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	LittleTether
Ticker/Symbol	LITE
Blockchain	Binance Smart Chain BEP20
Contract Address	0x12bF34d7418A9025d6B3a54e42b993c431224b51
Creator Address	0xCB0961C82E2Cae006B4722414e773fAC4CCDC76c
Current Owner Address	0xCB0961C82E2Cae006B4722414e773fAC4CCDC76c
Contract Explorer	https://bscscan.com/token/0x12bF34d7418A9025d6B3a54e42b993c431224b51#code
Compiler Version	v0.8.7+commit.e28d00a7
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	10,000,000 \$LITE
Decimals	9




Creation/Audit

Contract Deployed	26.02.2024
Audit Created	14.03.2024
Audit Update	V 1.0







Verified Socials



Website	https://littletether.com
Telegram	https://t.me/LittleTetherBsc
Twitter (X)	https://x.com/LittletetherBsc

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		0xCB0961C82E2Cae006B4722414e773fAC4CCDC76c
Buy Tax	10 %	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	10 %	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
Liquidity Status		Liquidity Status on 13.03.2024: Locked: 99.85% Mudra Locker for 78 days.
Trading Disable Functions		Trading suspendable function found. 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. The contract owner ay 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		<p>No Balance Modifier function found.</p> <p>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.</p>
Blacklist Function		<p>Blacklist Setting function found.</p> <p>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.</p>
Whitelist Function		<p>Whitelist Setting function found.</p> <p>If there is a function for this Developer can set zero fee or no max wallet size for addresses (for example team wallets can trade without fee. Can cause farming)</p>
Hidden Owner Analysis		<p>Hidden or multi owner with authorisations</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.</p>
Retrieve Ownership Function		<p>No Functions found which can retrieve ownership of the contract.</p> <p>If this function exists, it is possible for the project owner to regain ownership even after relinquishing it. Also known as fake renounce.</p>
Self Destruct Function		<p>No Self Destruct function found.</p> <p>If this function exists and is triggered, the contract will be destroyed, all functions will be unavailable, and all related assets will be erased.</p>
Specific Tax Changing Function		<p>No Specific Tax Changing Functions found.</p> <p>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!</p>
Trading Cooldown Function		<p>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.</p>
Max Transaction and Holding Modify Function		<p>Max Transaction and Holding Modify function found</p> <p>If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot</p>
Transaction Limiting Function		<p>No Transaction Limiter Function Found.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>

Details of Risk - Attention Items

⚠️ Whitelist [Exclude wallets]

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

```

787     ftrace | funcSig
788     function setIsFeeExempt(address holder!, bool exempt!) external authorized {
789         |   isFeeExempt[holder!] = exempt!;
790     }

791     ftrace | funcSig
792     function setIsTxLimitExempt(address holder!, bool exempt!) external authorized {
793         |   isTxLimitExempt[holder!] = exempt!;
794     }

795     ftrace | funcSig
796     function setIsTimelockExempt(address holder!, bool exempt!) external authorized {
797         |   isTimelockExempt[holder!] = exempt!;

```

⚠️ Max Transaction and Holding Modify Function

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

```

769     ftrace | funcSig
770     function setBuyTxLimitInPercent(uint256 maxBuyTxPercent!) external authorized {
771         |   maxBuyTxAmount = totalSupply.mul(maxBuyTxPercent!).div(10000);
772     }

773     ftrace | funcSig
774     function setSellTxLimitInPercent(uint256 maxSellTxPercent!) external authorized {
775         |   maxSellTxAmount = totalSupply.mul(maxSellTxPercent!).div(10000);
776     }

481     ftrace | funcSig
482     function setMaxWalletPercent(uint256 maxWallPercent!) external onlyOwner() {
483         |   maxWalletToken = totalSupply.mul(maxWallPercent!).div(10000);
484     }
485
486

```

⚠ Set Fee

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).

```

799  function setBuyFees(uint256 _liquidityFeeBuy!, uint256 _buybackFeeBuy!, uint256 _reflectionFeeBuy!, uint256 _marketingFeeBuy!, uint256 _devFeeBuy!, uint256 _feeDenominator!) external authorized {
800      liquidityFeeBuy = _liquidityFeeBuy!;
801      buybackFeeBuy = _buybackFeeBuy!;
802      reflectionFeeBuy = _reflectionFeeBuy!;
803      marketingFeeBuy = _marketingFeeBuy!;
804      devFeeBuy = _devFeeBuy!;
805      totalFeeBuy = _liquidityFeeBuy!.add(_buybackFeeBuy!).add(_reflectionFeeBuy!).add(_marketingFeeBuy!).add(_devFeeBuy!);
806      feeDenominator = _feeDenominator!;
807  }
808
809  function setSellFees(uint256 _liquidityFeeSell!, uint256 _buybackFeeSell!, uint256 _reflectionFeeSell!, uint256 _marketingFeeSell!, uint256 _devFeeSell!, uint256 _feeDenominator!) external authorized {
810      liquidityFeeSell = _liquidityFeeSell!;
811      buybackFeeSell = _buybackFeeSell!;
812      reflectionFeeSell = _reflectionFeeSell!;
813      marketingFeeSell = _marketingFeeSell!;
814      devFeeSell = _devFeeSell!;
815      totalFeeSell = _liquidityFeeSell!.add(_buybackFeeSell!).add(_reflectionFeeSell!).add(_marketingFeeSell!).add(_devFeeSell!);
816      feeDenominator = _feeDenominator!;
817  }
818
  
```

⚠ Blacklist

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.

```

649  function blacklistAddress(address _address!, bool _value!) public authorized{
650      isBlacklisted[_address!] = _value!;
651  }
652
  
```

⚠ Trading Cooldown

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

```

643
644  function cooldownEnabled(bool _status!, uint8 _interval!) public authorized {
645      buyCooldownEnabled = _status!;
646      cooldownTimerInterval = _interval!;
647  }
648
  
```


Hidden or multi owner with authorisations

For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned.

```
70 | ftrace
71 | constructor(address owner!) {
72 |     owner = owner!;
73 |     authorizations[owner!] = true;
74 | }
75 |
76 | modifier onlyOwner() {
77 |     require(isOwner(msg.sender), "!OWNER"); _;
78 | }
79 |
80 | modifier authorized() {
81 |     require(isAuthorized(msg.sender), "!AUTHORIZED"); _;
82 | }
83 |
84 | ftrace | funcSig
85 | function authorize(address account!) public onlyOwner {
86 |     authorizations[account!] = true;
87 | }
88 |
89 | ftrace | funcSig
90 | function unauthorize(address account!) public onlyOwner {
91 |     authorizations[account!] = false;
92 | }
93 |
94 | ftrace | funcSig
95 | function isOwner(address account!) public view returns (bool) {
96 |     return account! == owner;
97 | }
98 |
99 | ftrace | funcSig
100 | function isAuthorized(address account!) public view returns (bool) {
101 |     return authorizations[account!];
102 | }
103 |
104 | ftrace | funcSig
105 | function transferOwnership(address payable account!) public onlyOwner {
106 |     owner = account!;
107 |     authorizations[account!] = true;
108 |     emit OwnershipTransferred(account!);
109 | }
110 |
111 | event OwnershipTransferred(address owner);
112 | }
```

Contract Security

Total Findings: 5



■ **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 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

 **High severity Issues: (0)**

 **Medium severity issues: (0)**

 **Low severity issues: (4)**

- Long number literals
- Missing Events
- Unchecked Array Lenght
- Outdated Compiler Version

 **Informational severity issues: (1)**

- Public Functions Should be Declared External

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	Low	Low
SWC-103	Floating Pragma	Passed	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

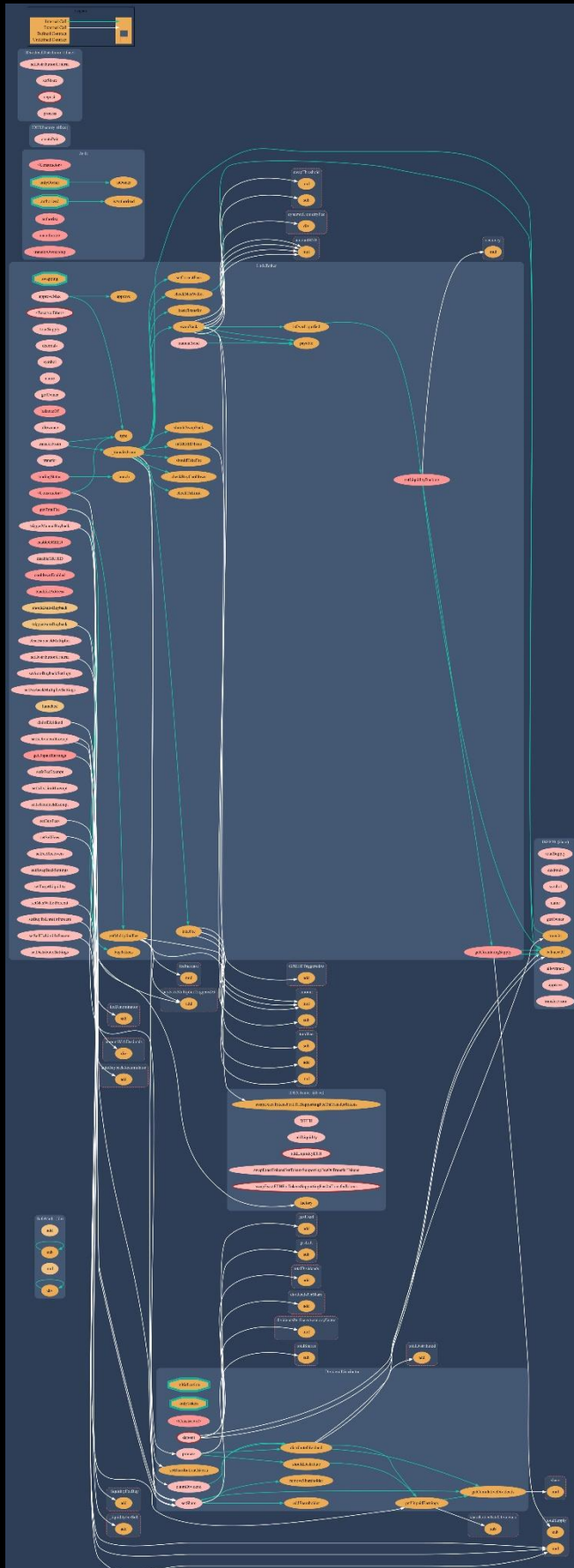
Detected High and Medium Severity Vulnerability Description.

⚠ Outdated Compiler Version (1 Item)

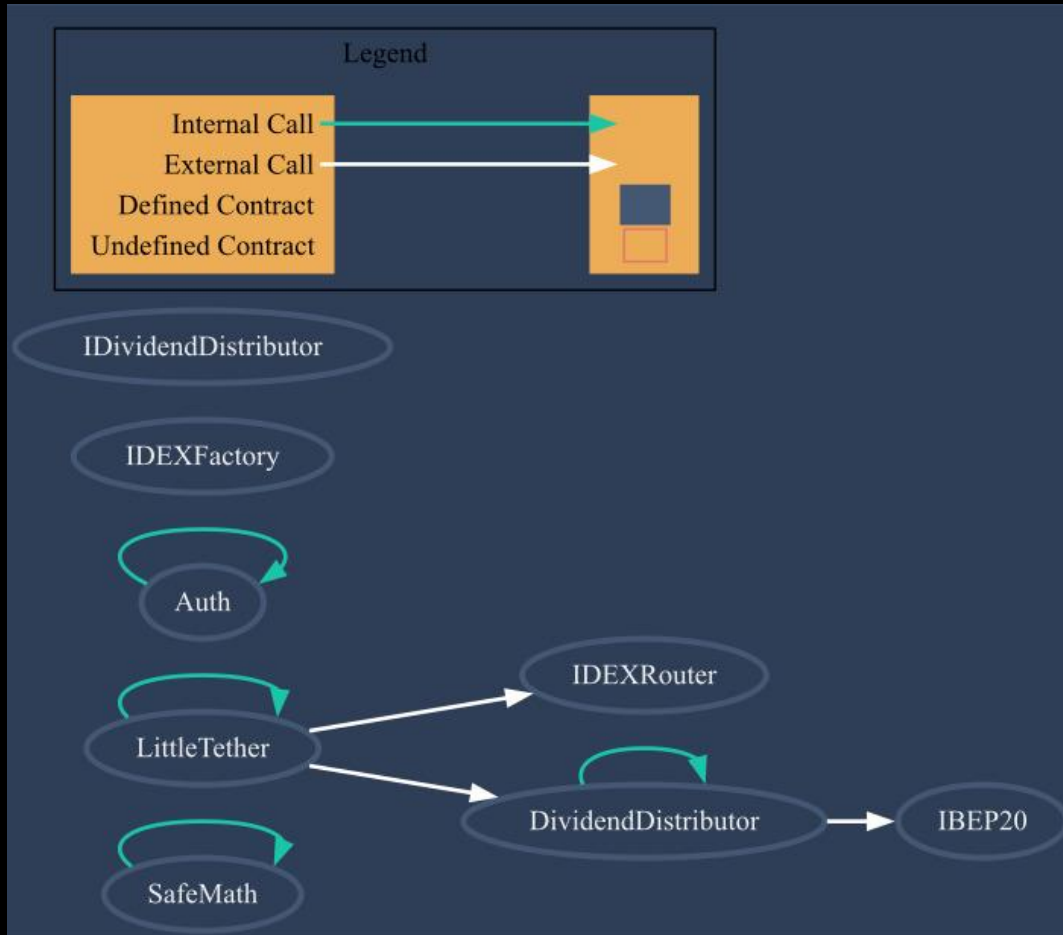
Item: 1	Location:	Line 11	Severity:	■ Low
---------	-----------	---------	-----------	-------

Function	Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version. The following outdated versions were detected: /little.sol - ^0.8.7
Remedation	It is recommended to use a recent version of the Solidity compiler that should not be the most recent version, and it should not be an outdated version as well. Using very old versions of Solidity prevents the benefits of bug fixes and newer security checks. Consider using the solidity version v0.8.23, which patches most solidity vulnerabilities.

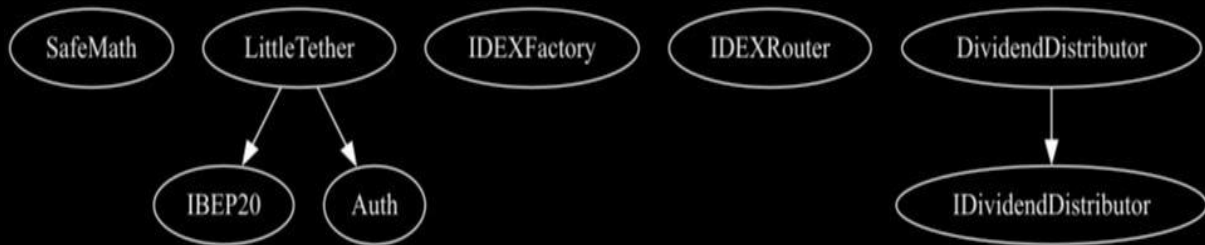
Contract Flow Graph




























Contract Interaction Graph








Inheritance Graph

























Contract Functions

























Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
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 
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	mod	Internal 		
Address	Library			














L	isContract	Internal 🔒		
L	sendValue	Internal 🔒	🔒	
L	functionCall	Internal 🔒	🔒	
L	functionCall	Internal 🔒	🔒	
L	functionCallWithValue	Internal 🔒	🔒	
L	functionCallWithValue	Internal 🔒	🔒	
L	_functionCallWithValue	Private 🔒	🔒	
Ownable	Implementation	Context		
L		Public 🔒	🔒	NO 🔒
L	owner	Public 🔒		NO 🔒
L	transferOwnership	Public 🔒	🔒	onlyOwner
L	getTime	Public 🔒		NO 🔒
IUniswapV2Factory	Interface			
L	feeTo	External 🔒		NO 🔒
L	feeToSetter	External 🔒		NO 🔒
L	getPair	External 🔒		NO 🔒
L	allPairs	External 🔒		NO 🔒
L	allPairsLength	External 🔒		NO 🔒
L	createPair	External 🔒	🔒	NO 🔒
L	setFeeTo	External 🔒	🔒	NO 🔒
L	setFeeToSetter	External 🔒	🔒	NO 🔒
IUniswapV2Pair	Interface			

L	name	External ¶		NO ¶
L	symbol	External ¶		NO ¶
L	decimals	External ¶		NO ¶
L	totalSupply	External ¶		NO ¶
L	balanceOf	External ¶		NO ¶
L	allowance	External ¶		NO ¶
L	approve	External ¶		NO ¶
L	transfer	External ¶		NO ¶
L	transferFrom	External ¶		NO ¶
L	DOMAIN_SEPARATOR	External ¶		NO ¶
L	PERMIT_TYPEHASH	External ¶		NO ¶
L	nonces	External ¶		NO ¶
L	permit	External ¶		NO ¶
L	MINIMUM_LIQUIDITY	External ¶		NO ¶
L	factory	External ¶		NO ¶
L	token0	External ¶		NO ¶
L	token1	External ¶		NO ¶
L	getReserves	External ¶		NO ¶
L	price0CumulativeLast	External ¶		NO ¶
L	price1CumulativeLast	External ¶		NO ¶
L	kLast	External ¶		NO ¶
L	burn	External ¶		NO ¶

L	swap	External !		NO !
L	skim	External !		NO !
L	sync	External !		NO !
L	initialize	External !		NO !
IUniswapV2Router01	Interface			
L	factory	External !		NO !
L	WETH	External !		NO !
L	addLiquidity	External !		NO !
L	addLiquidityETH	External !		NO !
L	removeLiquidity	External !		NO !
L	removeLiquidityETH	External !		NO !
L	removeLiquidityWithPermit	External !		NO !
L	removeLiquidityETHWithPermit	External !		NO !
L	swapExactTokensForTokens	External !		NO !
L	swapTokensForExactTokens	External !		NO !
L	swapExactETHForTokens	External !		NO !
L	swapTokensForExactETH	External !		NO !
L	swapExactTokensForETH	External !		NO !
L	swapETHForExactTokens	External !		NO !
L	quote	External !		NO !

L	getAmountOut	External ¶		NO ¶
L	getAmountIn	External ¶		NO ¶
L	getAmountsOut	External ¶		NO ¶
L	getAmountsIn	External ¶		NO ¶
IUniswapV2Router02	Interface	IUniswapV2Router01		
L	removeLiquidityETHSupportingFeeOnTransferTokens	External ¶		NO ¶
L	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ¶		NO ¶
L	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ¶		NO ¶
L	swapExactETHForTokensSupportingFeeOnTransferTokens	External ¶		NO ¶
L	swapExactTokensForETHSupportingFeeOnTransferTokens	External ¶		NO ¶
Synchronix	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	increaseAllowance	Public 		NO 
L	decreaseAllowance	Public 		NO 
L	minimumTokensBeforeSwapAmount	Public 		NO 
L	approve	Public 		NO 
L	_approve	Private 		
L	setMarketPairStatus	Public 		onlyOwner
L	setIsTxLimitExempt	External 		onlyOwner
L	setIsExcludedFromFee	Public 		onlyOwner
L	setBuyTaxes	External 		onlyOwner
L	setSellTaxes	External 		onlyOwner
L	setDistributionSettings	External 		onlyOwner
L	setMaxTxAmount	External 		onlyOwner
L	enableDisableWalletLimit	External 		onlyOwner
L	setIsWalletLimitExempt	External 		onlyOwner
L	setWalletLimit	External 		onlyOwner
L	setNumTokensBeforeSwap	External 		onlyOwner
L	setMarketingWalletAddress	External 		onlyOwner
L	setdevWalletAddress	External 		onlyOwner

L	setSwapAndLiquifyEnabled	Public 🔒		onlyOwner
L	setSwapAndLiquifyByLimitOnly	Public 🔒		onlyOwner
L	getCirculatingSupply	Public 🔒		NO 🔒
L	transferToAddressETH	Private 🔒		
L	changeRouterVersion	Public 🔒		onlyOwner
L		External 🔒		NO 🔒
L	transfer	Public 🔒		NO 🔒
L	transferFrom	Public 🔒		NO 🔒
L	_transfer	Private 🔒		
L	_basicTransfer	Internal 🔒		
L	swapAndLiquify	Private 🔒		lockTheSwap
L	swapTokensForEth	Private 🔒		
L	addLiquidity	Private 🔒		
L	takeFee	Internal 🔒		



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 vulnerabilities in the code. Findings getting reported and improvements getting suggested.

Automatic and Manual Review

We are using automated tools to scan functions and weaknesses 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>

