SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

BABY LTC

BEP 20

0x59FB9a377004CC8974009F72d94b4d801C58499B

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	BABY LTC
Ticker/Simbol	BLTC
Blockchain	Binance Smart Chain Bep20
Contract Address	0x59FB9a377004CC8974009F72d94b4d801C58499B
Creator Address	0xB811D56ed0E71794869A2445cb159B2C235e46c9
Current Owner Address	0xC3bc56601d96Faa84beE11EB587F4bF89F9Df529
Contract Explorer	https://bscscan.com/token/0x59FB9a377004CC8974009F72d94b4d801C58499B
Compiler Version	v0.8.4+commit.c7e474f2
License	MIT License
Optimisation	Yes with 200 Runs
Total Supply	1,000,000,000 BLTC
Decimals	18

Creation/Audit

Contract Deployed	5 May 2023
Audit Created	22-Aug-23 21:00:00 UTC
Audit Update	V 0.1

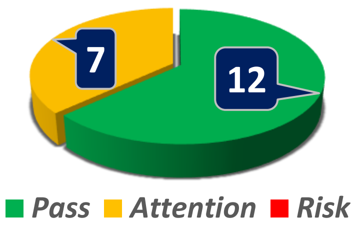
Verified Socials

Website	https://babyltc.io
Telegram	https://t.me/+rJXf3p1fjnljYWRk
Twitter	https://twitter.com/babyltc_

BABY LTC TOKEN BEP20

Contract Function Analysis

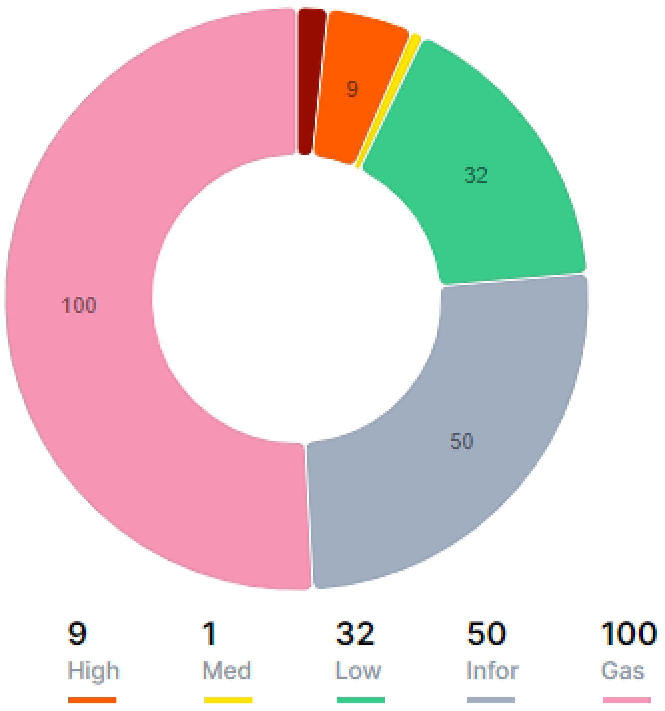
✓ Pass ⚠ Attention Item ⚠ Risky Item



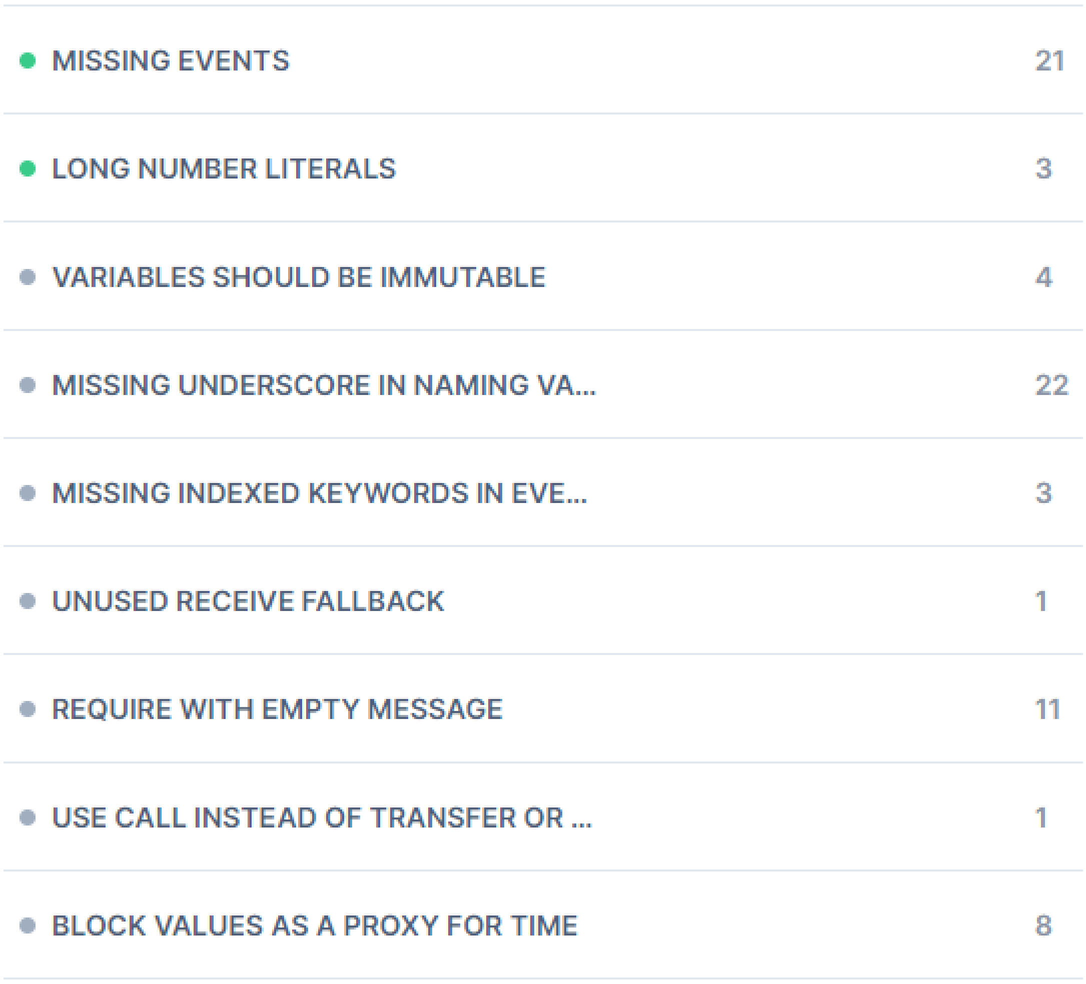
Contract Verified	✓	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership	✓	The ownership of the contract was sent to dead address. With this the owner eliminates he's rights to modify the contract. The owner can not set any of the functions anymore.
Buy Tax	14 %	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. **Setfee max found: 25%**
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. **Setfee max found: 25%**
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	✓	Locked on 22.08.2023: 62% for 43331 days on Mudra locker Burned (locked forewer) 38% Note! Initial liqudity tokens scanned. For new LP Lockers allways re-check with skeleton scanner on telegram.
Trading Disable Functions	✓	No 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. ⚠ If there is authorised hidden owner, or there is Retrieve Ownership Function, the trading disable function may be used!
Set Fees function	⚠	Fee Setting function found. 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). ⚠ If there is authorised hidden owner, or there is Retrieve Ownership Function, the set fees function may be used!
Proxy Contract	✓	The proxy contract means contract owner can modifiy the function of the token and possibly effect the price. The Owner is not the creator but the creator may have authorisation to change functions.
Mint Function	✓	No mint function found. 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. If contract is renounced this function can't be used.

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. ⚠ If contract is renounced this function still can be used as auto self Destruct
Whitelist Function	⚠	Whitelist Function Found. 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) If there is a whitelist, some addresses may not be able to trade normally (honeypot risk
Hidden Owner Analysis	✓	No authorised hidden owner found. For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.
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	⚠	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	⚠	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. If there is a function for this, the maximum trading amount or maximum position can be modified. Can cause honeypot
Transaction Limiting Function	⚠	Transaction Limiter Function Found. The number of overall token transactions may be limited (honeypot risk)

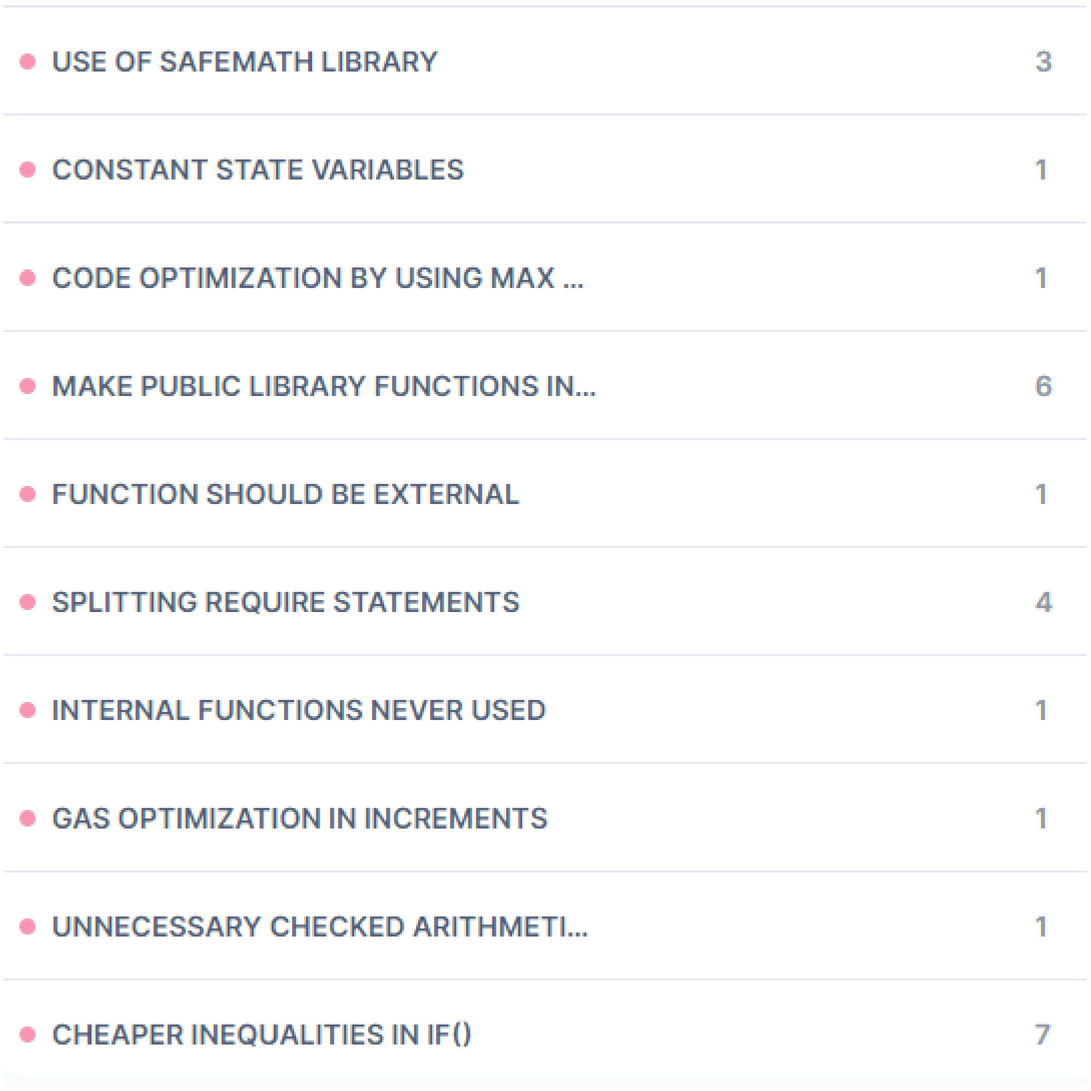
Contract Safety and Weakness



🟥	INCORRECT ACCESS CONTROL	3
🟧	USE OF TX.ORIGIN	2
🟧	UNCHECKED ARRAY LENGTH	1
🟧	UNCHECKED TRANSFER	3
🟧	APPROVE FRONT-RUNNING ATTACK	3
🟨	DANGEROUS UNARY EXPRESSIONS	1
🟩	USE OF MULTIPLE PRAGMA VERSIONS	1
🟩	USE OWNABLE2STEP	3
🟩	EMPTY TRY/CATCH BLOCK	3
🟩	OUTDATED COMPILER VERSION	1



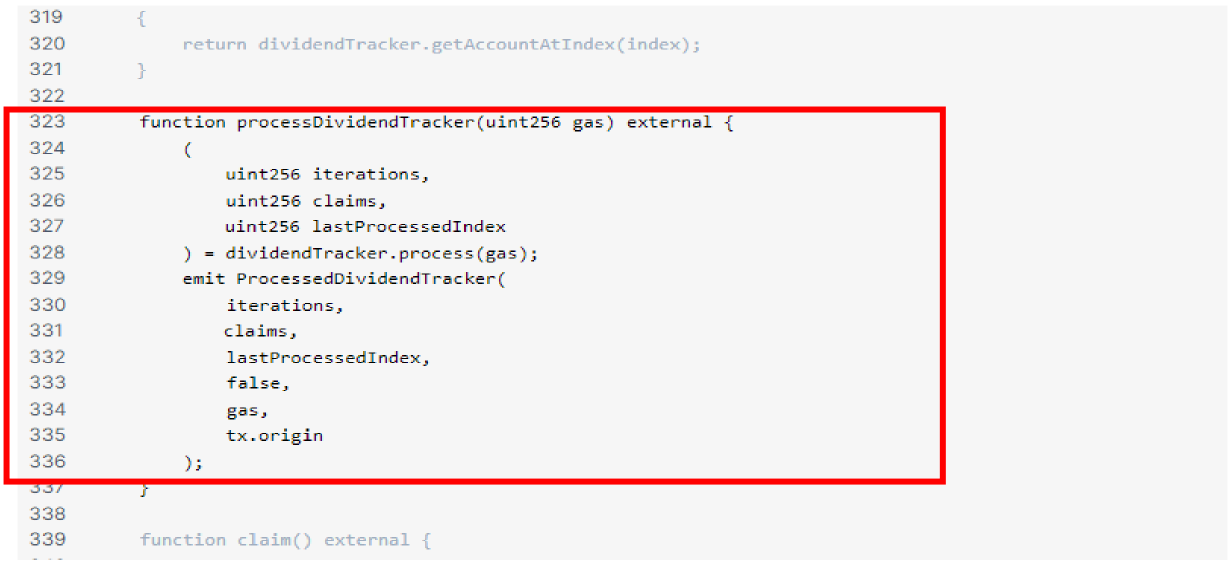
●	MISSING EVENTS	21
●	LONG NUMBER LITERALS	3
●	VARIABLES SHOULD BE IMMUTABLE	4
●	MISSING UNDERSCORE IN NAMING VA...	22
●	MISSING INDEXED KEYWORDS IN EVE...	3
●	UNUSED RECEIVE FALLBACK	1
●	REQUIRE WITH EMPTY MESSAGE	11
●	USE CALL INSTEAD OF TRANSFER OR ...	1
●	BLOCK VALUES AS A PROXY FOR TIME	8



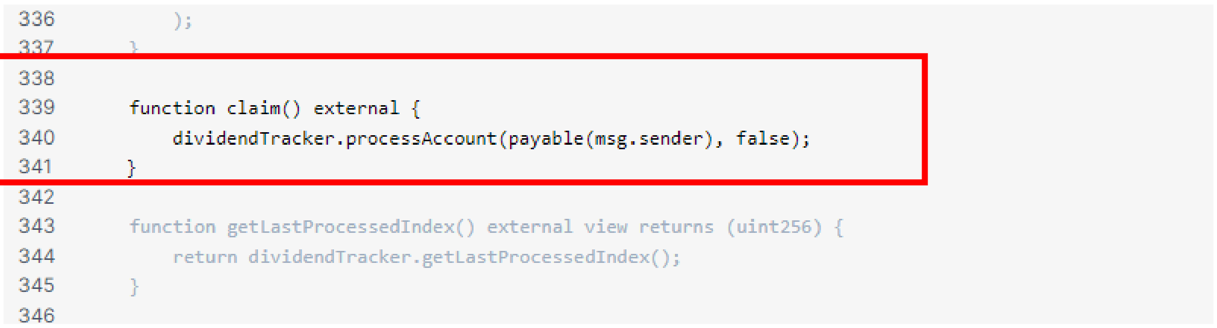
● USE OF SAFEMATH LIBRARY	3
● CONSTANT STATE VARIABLES	1
● CODE OPTIMIZATION BY USING MAX ...	1
● MAKE PUBLIC LIBRARY FUNCTIONS IN...	6
● FUNCTION SHOULD BE EXTERNAL	1
● SPLITTING REQUIRE STATEMENTS	4
● INTERNAL FUNCTIONS NEVER USED	1
● GAS OPTIMIZATION IN INCREMENTS	1
● UNNECESSARY CHECKED ARITHMETI...	1
● CHEAPER INEQUALITIES IN IF()	7

● FUNCTION SHOULD RETURN STRUCT	4
● ARRAY LENGTH CACHING	1
● VARIABLES DECLARED BUT NEVER US...	1
● PUBLIC CONSTANTS CAN BE PRIVATE	1
● LONG REQUIRE/REVERT STRINGS	11
● CHEAPER INEQUALITIES IN REQUIRE()	10
● CHEAPER CONDITIONAL OPERATORS	6
● STORAGE VARIABLE CACHING IN MEM...	40

⚠ Incorrect Acces Control (3 item)



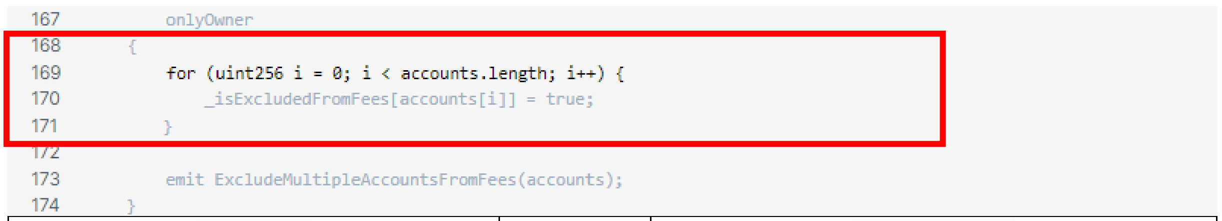
```
    {
        return dividendTracker.getAccountAtIndex(index);
    }

    function processDividendTracker(uint256 gas) external {
        (
            uint256 iterations,
            uint256 claims,
            uint256 lastProcessedIndex
        ) = dividendTracker.process(gas);
        emit ProcessedDividendTracker(
            iterations,
            claims,
            lastProcessedIndex,
            false,
            gas,
            tx.origin
        );
    }

    function claim() external {
```



```
        );
    }

    function claim() external {
        dividendTracker.processAccount(payable(msg.sender), false);
    }

    function getLastProcessedIndex() external view returns (uint256) {
        return dividendTracker.getLastProcessedIndex();
    }

```

Function	Severity	Remedation
Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract. The contract BABYTOKEN is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function processDividendTracker is missing the modifier onlyOwner.	Severity : Critical	It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

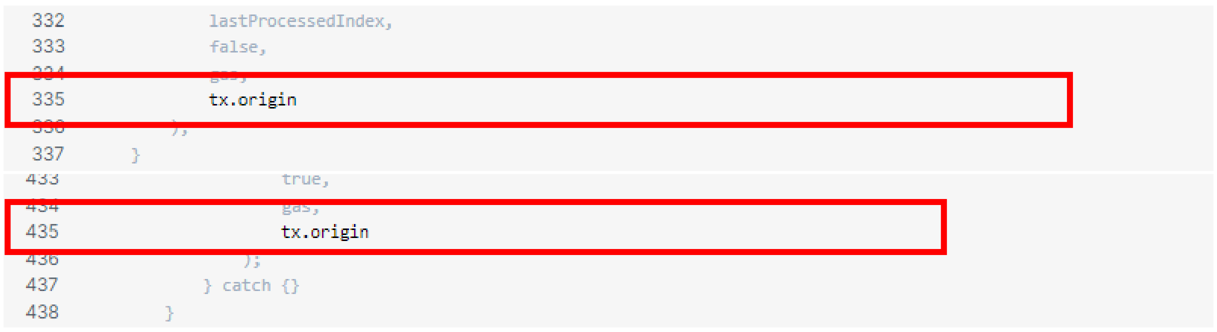
Unchecked Array Lenght (1 item)



```
        onlyOwner
    {
        for (uint256 i = 0; i < accounts.length; i++) {
            _isExcludedFromFees[accounts[i]] = true;
        }

        emit ExcludeMultipleAccountsFromFees(accounts);
    }
```

Function	Severity	Remedation
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++) { cosltyFunc(); } 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.	Severity : High	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.

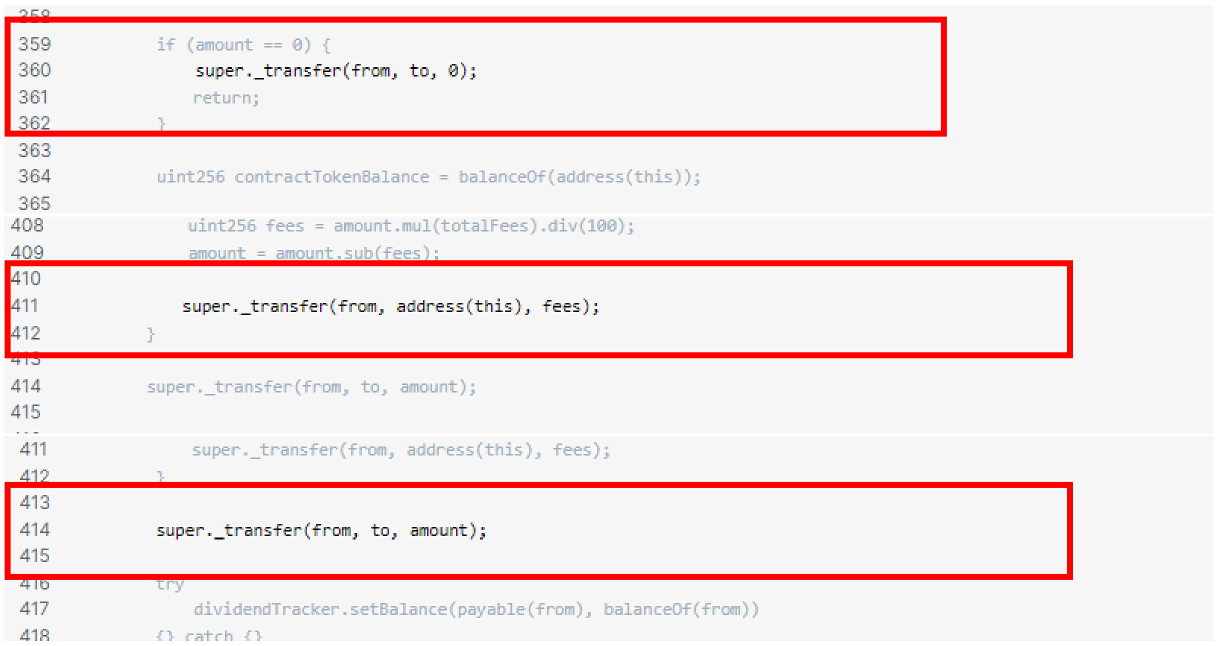
⚠ Use of TX. origin (2 Item)



```
                lastProcessedIndex,
                false,
                gas,
                tx.origin
            );
        }
                        true,
                        gas,
                        tx.origin
                    );
                } catch {}
            }
```

Function	Severity	Remedation
In Solidity, tx.origin is a global variable that returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable. For example, if an authorized account calls a malicious contract which triggers it to call the vulnerable contract that passes an authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.	Severity : High	tx.origin should not be used for authorization in smart contracts. It does have some legitimate use cases, for example, To prevent external contracts from calling the current contract, you can implement a require of the form require(tx.origin == msg.sender). This prevents intermediate contracts from calling the current contract, thus limiting the contract to regular codeless addresses.

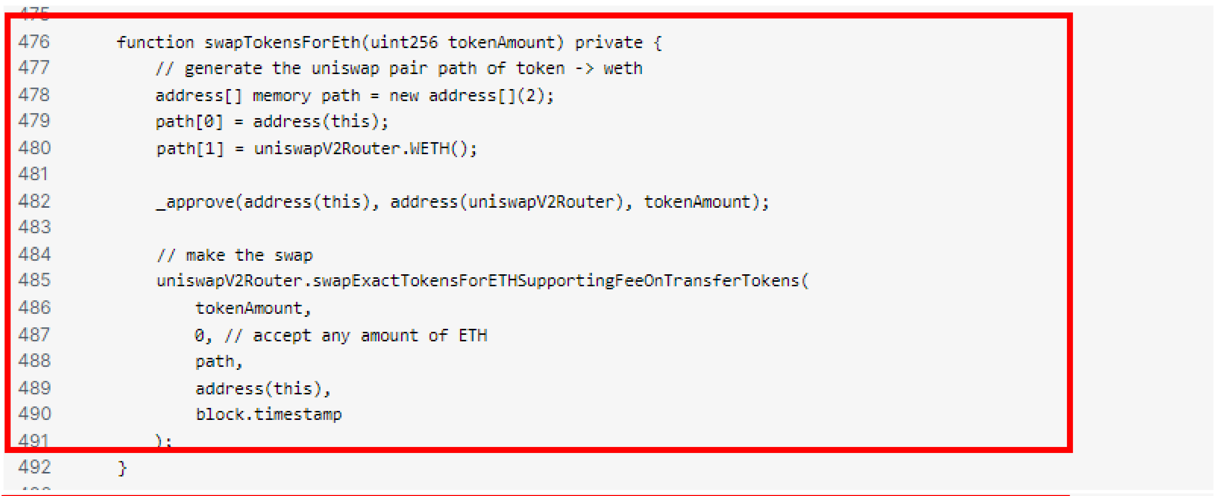
⚠ Unchecked Transfers (3 Item)



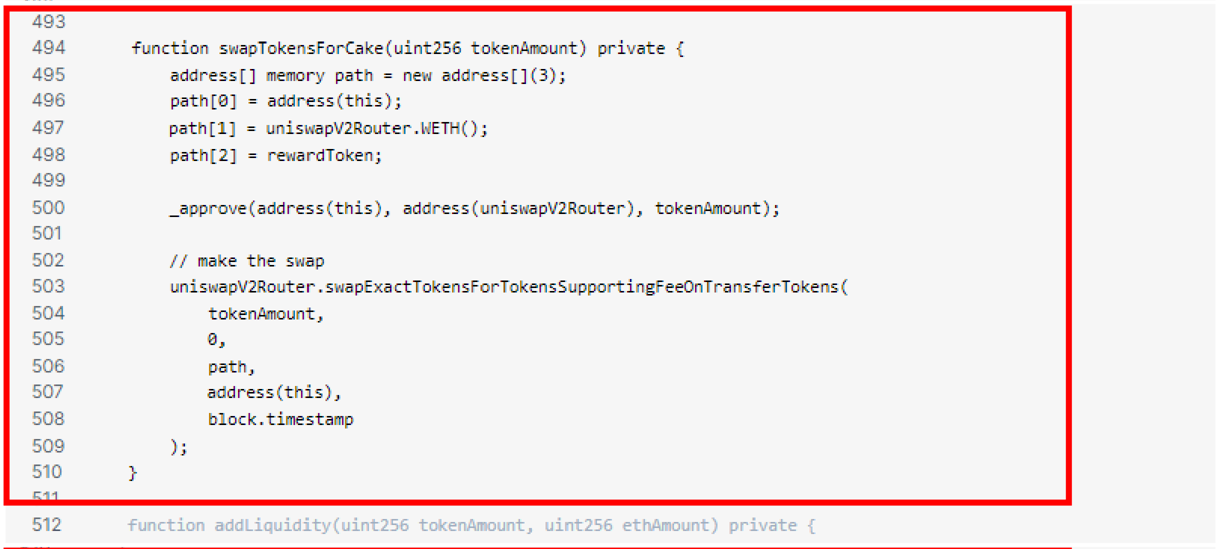
```
        if (amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

            uint256 fees = amount.mul(totalFees).div(100);
            amount = amount.sub(fees);

            super._transfer(from, address(this), fees);
        }

        super._transfer(from, to, amount);

            super._transfer(from, address(this), fees);
        }

        super._transfer(from, to, amount);

        try
            dividendTracker.setBalance(payable(from), balanceOf(from))
        {} catch {}
```

Function	Severity	Remedation
Some tokens do not revert the transaction when the transfer or transferFrom fails and returns False. Hence we must check the return value after calling the transfer or transferFrom function.	Severity : High	Use OpenZeppelin SafeERC20's safetransfer and safetransferFrom functions.

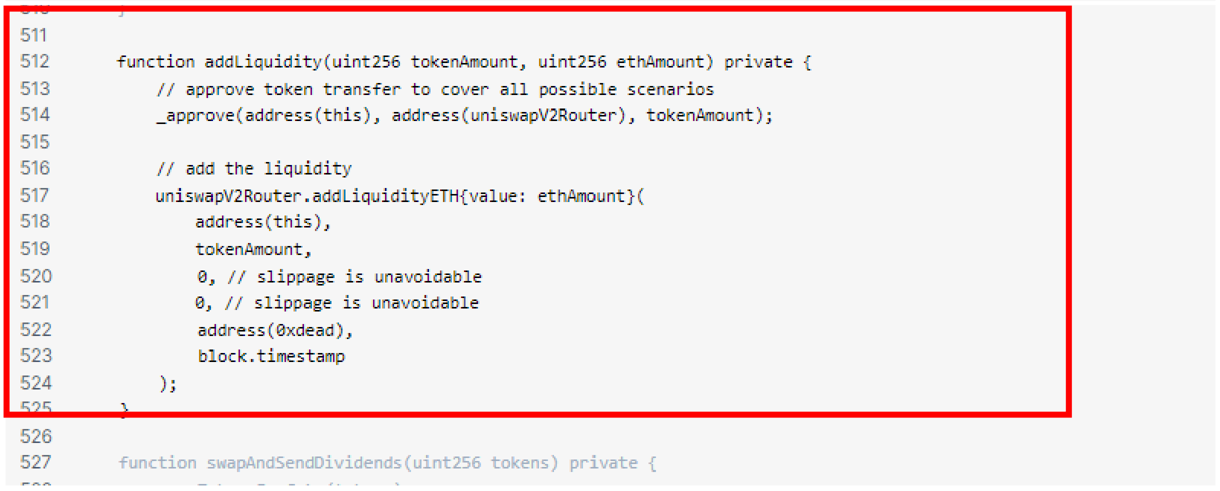
⚠ Approve Front Running Attack (3 Item)



```
    function swapTokensForEth(uint256 tokenAmount) private {
        // generate the uniswap pair path of token -> weth
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();

        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // make the swap
        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );
    }
```



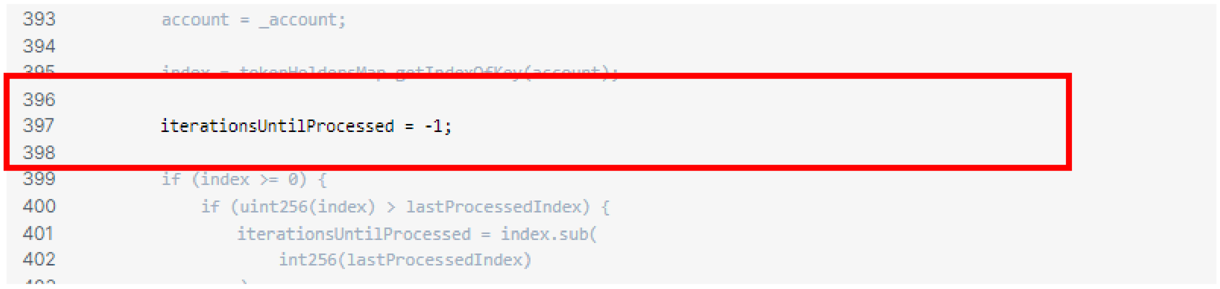
```
    function swapTokensForCake(uint256 tokenAmount) private {
        address[] memory path = new address[](3);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        path[2] = rewardToken;

        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // make the swap
        uniswapV2Router.swapExactTokensForTokensSupportingFeeOnTransferTokens(
            tokenAmount,
            0,
            path,
            address(this),
            block.timestamp
        );
    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
```



```
    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            address(0xdead),
            block.timestamp
        );
    }

    function swapAndSendDividends(uint256 tokens) private {
```

Function	Severity	Remedation
The swapTokensForEth() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when they try to withdraw certain tokens from the sender's account. Meanwhile, if the sender decides to change the amount and sends another approve transaction, the receiver can notice this transaction before it's mined and can extract tokens from both the transactions, therefore, ending up with tokens from both the transactions. This is a front-running attack affecting the ERC20 Approve function. The function swapTokensForEth can be front-run by abusing the _approve function.	Severity : High	Only use the approve function of the ERC/BEP standard to change the allowed amount to 0 or from 0 (wait till transaction is mined and approved). Token owner just needs to make sure that the first transaction actually changed allowance from N to 0, i.e., that the spender didn't manage to transfer some of N allowed tokens before the first transaction was mined. Such checking is possible using advanced blockchain explorers such as [Etherscan.io](https://etherscan.io/) Another way to mitigate the threat is to approve token transfers only to smart contracts with verified source code that does not contain logic for performing attacks like described above, and to accounts owned by the people you may trust.

⚠ Dangerous Unary Expressions (1 Item)

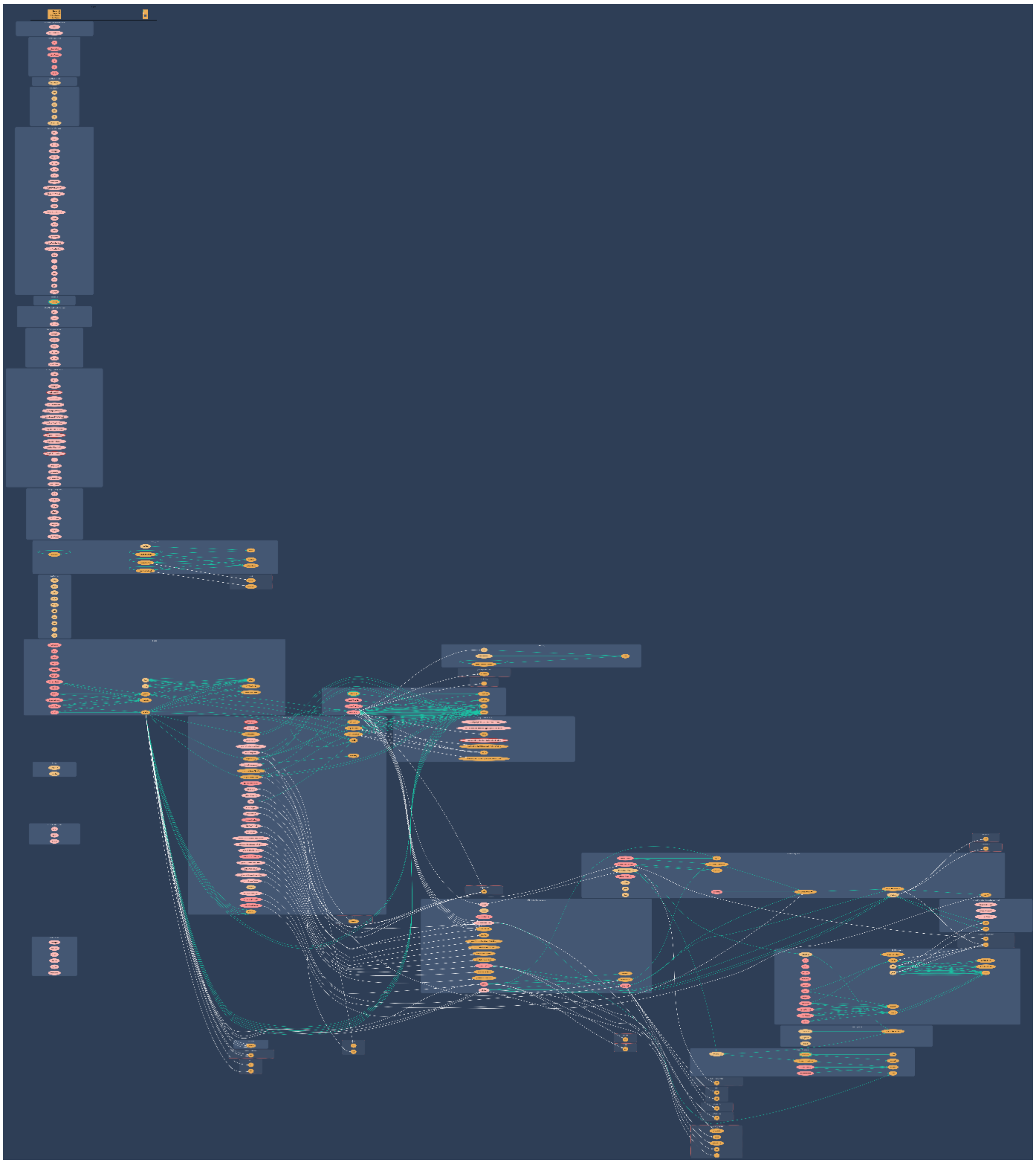


```
        account = _account;

        index = tokenHoldersMap.getIndexOfKey(account);

        iterationsUntilProcessed = -1;

        if (index >= 0) {
            if (uint256(index) > lastProcessedIndex) {
                iterationsUntilProcessed = index.sub(
                    int256(lastProcessedIndex)
                );
```

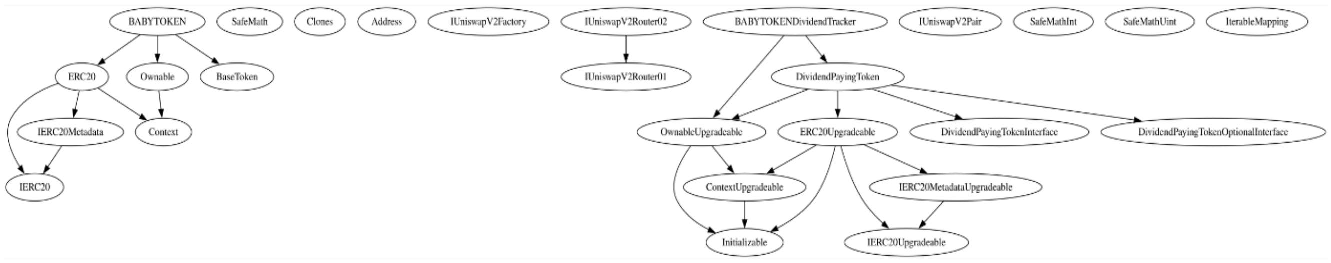
Function	Severity	Remedation

Mathematical operations in the smart contracts forms the base for all the arithmetic logic in the code. Care should be taken when implementing these because they may control critical function logic, tokens, ether, etc.
The contract was found to be incorrectly implementing arithmetic expressions assuming the contract wanted to add the value of a to b and store it back in a.
Correct usage: a += b;
Incorrect usage: a =+b;
	Severity : medium	Developers should exercise caution when writing these expressions because a simple mistake may cause loss of funds or compromise of the contract.
		Make sure that the expressions used are in the correct format, i.e., a += b; and not a =+b;.

Contract Flow Graph



Inheritance Graph



Contract Descriptions

Contract	File	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External ❗		NO❗
	balanceOf	External ❗		NO❗
	transfer	External ❗	🛑	NO❗
	allowance	External ❗		NO❗
	approve	External ❗	🛑	NO❗
	transferFrom	External ❗	🛑	NO❗
IERC20Metadata	Interface	IERC20		
	name	External ❗		NO❗
	symbol	External ❗		NO❗
	decimals	External ❗		NO❗
Context	Implementation			
	_msgSender	Internal 🔒		
	_msgData	Internal 🔒		
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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 🔒	🛑	
	_beforeTokenTransfer	Internal 🔒	🛑	
	_afterTokenTransfer	Internal 🔒	🛑	
Ownable	Implementation	Context		
		Public ❗	🛑	NO❗
	owner	Public ❗		NO❗
	renounceOwnership	Public ❗	🛑	onlyOwner
	transferOwnership	Public ❗	🛑	onlyOwner
	_setOwner	Private 🔐	🛑	
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 🔒		
Clones	Library			
	clone	Internal 🔒	🛑	
	cloneDeterministic	Internal 🔒	🛑	
	predictDeterministicAddress	Internal 🔒		

	predictDeterministicAddress	Internal 🔒		
Address	Library			
	isContract	Internal 🔒		
	sendValue	Internal 🔒	🛑	
	functionCall	Internal 🔒	🛑	
	functionCall	Internal 🔒	🛑	
	functionCallWithValue	Internal 🔒	🛑	
	functionCallWithValue	Internal 🔒	🛑	
	functionStaticCall	Internal 🔒		
	functionStaticCall	Internal 🔒		
	functionDelegateCall	Internal 🔒	🛑	
	functionDelegateCall	Internal 🔒	🛑	
	verifyCallResult	Internal 🔒		
IUniswapV2Factory	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❗
IUniswapV2Router01	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❗️
IUniswapV2Router02	Interface	IUniswapV2Router01		
	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗️	🛑	NO❗️
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗️	🛑	NO❗️
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗️	🛑	NO❗️
	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗️	💵	NO❗️

	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
IERC20Upgradeable	Interface			
	totalSupply	External ❗		NO❗
	balanceOf	External ❗		NO❗
	transfer	External ❗	🛑	NO❗
	allowance	External ❗		NO❗
	approve	External ❗	🛑	NO❗
	transferFrom	External ❗	🛑	NO❗
IERC20MetadataUpgradeable	Interface	IERC20Upgradeable		
	name	External ❗		NO❗
	symbol	External ❗		NO❗
	decimals	External ❗		NO❗
Initializable	Implementation			
ContextUpgradeable	Implementation	Initializable		
	__Context_init	Internal 🔒	🛑	initializer
	__Context_init_unchained	Internal 🔒	🛑	initializer
	_msgSender	Internal 🔒		
	_msgData	Internal 🔒		
ERC20Upgradeable	Implementation	Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable		
	__ERC20_init	Internal 🔒	🛑	initializer
	__ERC20_init_unchained	Internal 🔒	🛑	initializer

	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 🔒	🛑	
	_beforeTokenTransfer	Internal 🔒	🛑	
	_afterTokenTransfer	Internal 🔒	🛑	
OwnableUpgradeable	Implementation	Initializable, ContextUpgradeable		
	__Ownable_init	Internal 🔒	🛑	initializer
	__Ownable_init_unchained	Internal 🔒	🛑	initializer
	owner	Public ❗		NO❗
	renounceOwnership	Public ❗	🛑	onlyOwner
	transferOwnership	Public ❗	🛑	onlyOwner
	_setOwner	Private 🔐	🛑	
IUniswapV2Pair	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❗
SafeMathInt	Library			
	mul	Internal 🔒		
	div	Internal 🔒		
	sub	Internal 🔒		
	add	Internal 🔒		
	abs	Internal 🔒		
	toUint256Safe	Internal 🔒		
SafeMathUint	Library			
	toInt256Safe	Internal 🔒		
IterableMapping	Library			
	get	Public ❗		NO❗

	getIndexOfKey	Public ❗		NO❗
	getKeyAtIndex	Public ❗		NO❗
	size	Public ❗		NO❗
	set	Public ❗	🛑	NO❗
	remove	Public ❗	🛑	NO❗
DividendPayingTokenInterface	Interface			
	dividendOf	External ❗		NO❗
	withdrawDividend	External ❗	🛑	NO❗
DividendPayingTokenOptionalInterface	Interface			
	withdrawableDividendOf	External ❗		NO❗
	withdrawnDividendOf	External ❗		NO❗
	accumulativeDividendOf	External ❗		NO❗
DividendPayingToken	Implementation	ERC20Upgradeable, OwnableUpgradeable, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
	__DividendPayingToken_init	Internal 🔒	🛑	initializer
	distributeCAKEDividends	Public ❗	🛑	onlyOwner
	withdrawDividend	Public ❗	🛑	NO❗
	_withdrawDividendOfUser	Internal 🔒	🛑	

	dividendOf	Public ❗		NO❗
	withdrawableDividendOf	Public ❗		NO❗
	withdrawnDividendOf	Public ❗		NO❗
	accumulativeDividendOf	Public ❗		NO❗
	_transfer	Internal 🔒	🛑	
	_mint	Internal 🔒	🛑	
	_burn	Internal 🔒	🛑	
	_setBalance	Internal 🔒	🛑	
BABYTOKENDividendTracker	Implementation	OwnableUpgradeable, DividendPayingToken		
	initialize	External ❗	🛑	initializer
	_transfer	Internal 🔒		
	withdrawDividend	Public ❗		NO❗
	excludeFromDividends	External ❗	🛑	onlyOwner
	isExcludedFromDividends	Public ❗		NO❗
	updateClaimWait	External ❗	🛑	onlyOwner
	updateMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
	getLastProcessedIndex	External ❗		NO❗
	getNumberOfTokenHolders	External ❗		NO❗
	getAccount	Public ❗		NO❗
	getAccountAtIndex	Public ❗		NO❗
	canAutoClaim	Private 🔐		
	setBalance	External ❗	🛑	onlyOwner
	process	Public ❗	🛑	NO❗
	processAccount	Public ❗	🛑	onlyOwner

BaseToken	Implementation			
BABYTOKEN	Implementation	ERC20, Ownable, BaseToken		
		Public ❗	💵	ERC20
		External ❗	💵	NO❗
	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
	excludeFromFees	External ❗	🛑	onlyOwner
	excludeMultipleAccountsFromFees	External ❗	🛑	onlyOwner
	setMarketingWallet	External ❗	🛑	onlyOwner
	setTokenRewardsFee	External ❗	🛑	onlyOwner
	setLiquiditFee	External ❗	🛑	onlyOwner
	setMarketingFee	External ❗	🛑	onlyOwner
	_setAutomatedMarketMakerPair	Private 🔐	🛑	
	updateGasForProcessing	Public ❗	🛑	onlyOwner
	updateClaimWait	External ❗	🛑	onlyOwner
	getClaimWait	External ❗		NO❗
	updateMinimumTokenBalanceForDividends	External ❗	🛑	onlyOwner
	getMinimumTokenBalanceForDividends	External ❗		NO❗
	getTotalDividendsDistributed	External ❗		NO❗
	isExcludedFromFees	Public ❗		NO❗
	withdrawableDividendOf	Public ❗		NO❗
	dividendTokenBalanceOf	Public ❗		NO❗
	excludeFromDividends	External ❗	🛑	onlyOwner

	isExcludedFromDividends	Public ❗		NO❗
	getAccountDividendsInfo	External ❗		NO❗
	getAccountDividendsInfoAtIndex	External ❗		NO❗
	processDividendTracker	External ❗	🛑	NO❗
	claim	External ❗	🛑	NO❗
	getLastProcessedIndex	External ❗		NO❗
	getNumberOfDividendTokenHolders	External ❗		NO❗
	_transfer	Internal 🔒	🛑	
	swapAndSendToFee	Private 🔐	🛑	
	swapAndLiquify	Private 🔐	🛑	
	swapTokensForEth	Private 🔐	🛑	
	swapTokensForCake	Private 🔐	🛑	
	addLiquidity	Private 🔐	🛑	
	swapAndSendDividends	Private 🔐	🛑	

🛑	Function can modify state	💵	Function is payable

Source:

File Name SHA-1 Hash

c:\Solidity\babyltc.sol b4e69c7c64c78595cd52c4f5dba4be96d2bb96c4

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