SKELETON ECOSYSTEM

SMART CONTRACT AUDIT

Baby BNB Miner (BBM)

BEP20

0xc91300841659cB080B0C966d7799d2060Cbc9

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	BNBMINER
Ticker/Simbol	BBM
Blockchain	Binance Smart Chain BEP20
Contract Address	0xc91300841659cB080B0C966d7799d2060Cbc9338
Creator Address	0xB613651ba9954e05b2e94D3c7BE9De4C60302B7a
Current Owner Address	0x0000000000000000000000000000000000000000
Contract Explorer	https://bscscan.com/address/0xc91300841659cB080B0C966d7799d2060Cbc9338#code
Compiler Version	v0.8.19+commit.7dd6d404
License	MIT
Optimisation	Yes with 200 Runs
Total Supply	1,000,000,000 **BBM**
Decimals	18

Creation/Audit

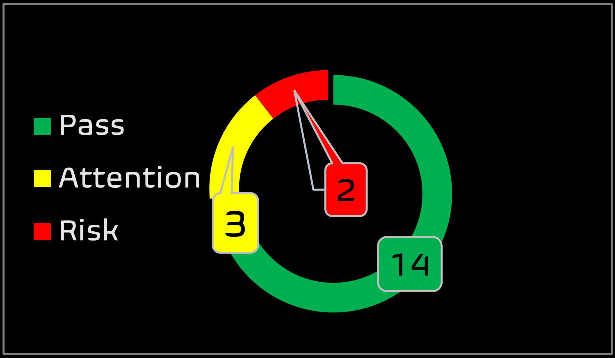
Contract Deployed	08.04.2024
Audit Created	09.03.2024
Audit Update	V 1.0

Verified Socials

Website	https://Babybnbminer.com
Telegram	https://t.me/BABYBNBMINER
Twitter (X)	https://twitter.com/Baby_BNB_MINER

Contract Function Analysis

✅ Pass ⚠️ Attention Item ❗ Risky Item

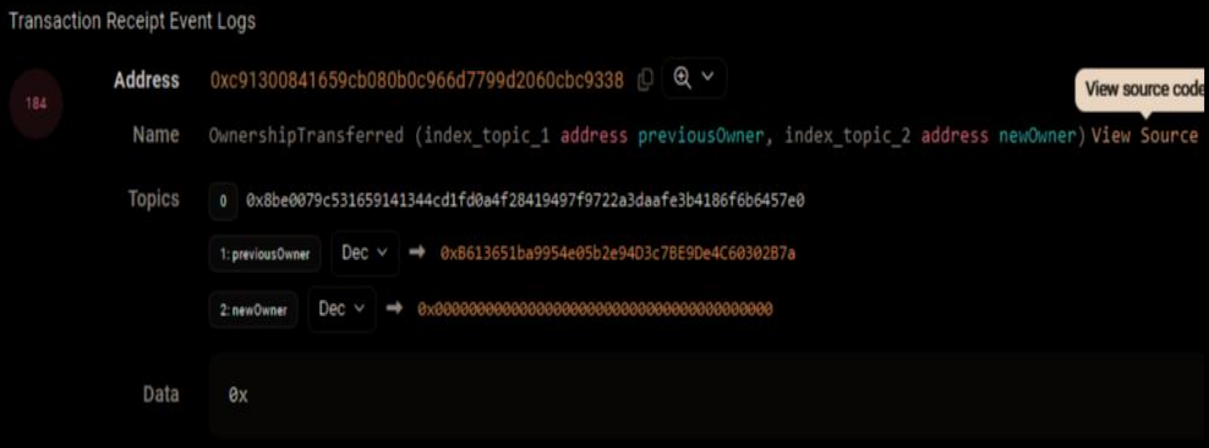


Contract Verified	✅	The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Ownership		0x0000000000000000000000000000000000000000 Sometimes referred to as the "zero address" or “dead address” and is not owned by anyone.
Buy Tax	10 %	Shows the 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
Liqudity Status	✅	Liqudity status on 13.04.2024 Lp Locked: 91% for 171 days on PinkSale
Trading Disable Functions	❗	Trading suspendable function found. Contract renounced, function can not be triggered by owner. 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, function can not be triggered by owner. 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 found. Contract renounced, function can not be triggered by owner. 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, function can not be triggered by owner. 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 with authorisation 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, function can not be triggered by owner. 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

Removing Risk of contract function based on renounced ownership



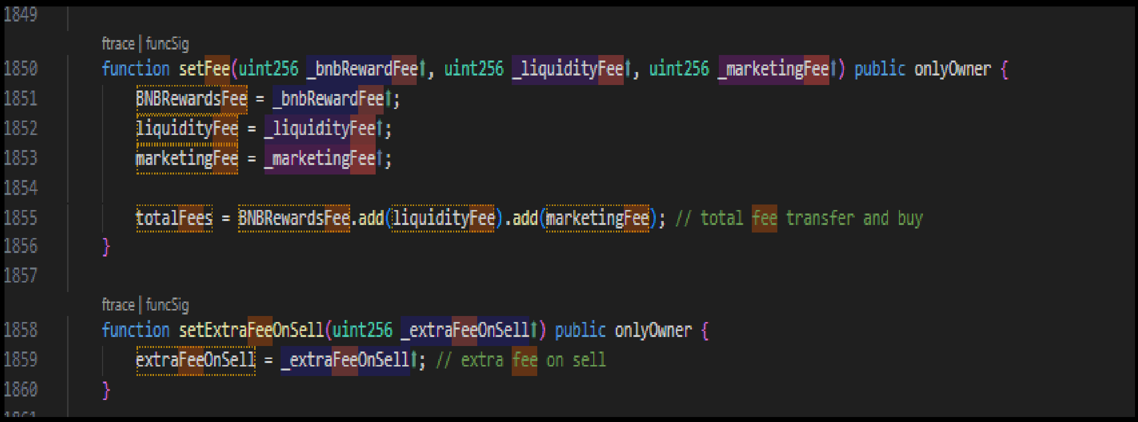
```
Transaction Receipt Event Logs

184   Address  0xc91300841659cb080b0c966d7799d2060cbc9338                View source code
      Name     OwnershipTransferred (index_topic_1 address previousOwner, index_topic_2 address newOwner) View Source
      Topics   0  0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0
               1: previousOwner  Dec → 0xB613651ba9954e05b2e94D3c7BE9De4C60302B7a
               2: newOwner  Dec → 0x0000000000000000000000000000000000000000
      Data     0x
```

Following detected contract functions serve as informational purposes about the contract. The owner has no more authorisation to trigger the following functions.

⚠ Set Fee

Contract renounced, function can not be triggered by owner.

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).

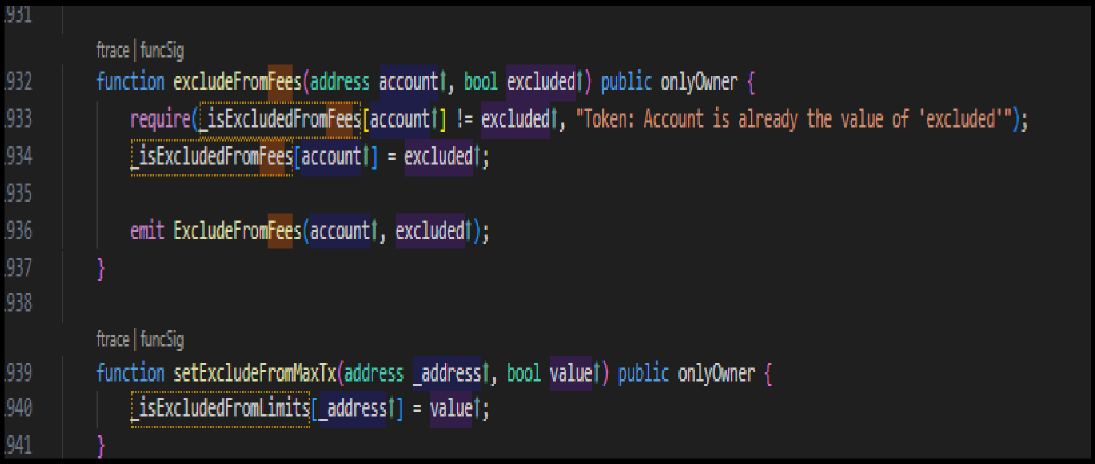


```
ftrace | funcSig
function setFee(uint256 _bnbRewardFee, uint256 _liquidityFee, uint256 _marketingFee) public onlyOwner {
    BNBRewardsFee = _bnbRewardFee;
    liquidityFee = _liquidityFee;
    marketingFee = _marketingFee;

    totalFees = BNBRewardsFee.add(liquidityFee).add(marketingFee); // total fee transfer and buy
}

ftrace | funcSig
function setExtraFeeOnSell(uint256 _extraFeeOnSell) public onlyOwner {
    extraFeeOnSell = _extraFeeOnSell; // extra fee on sell
}
```

⚠ Whitelist (Set exluded wallets)

Contract renounced, function can not be triggered by owner.

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)

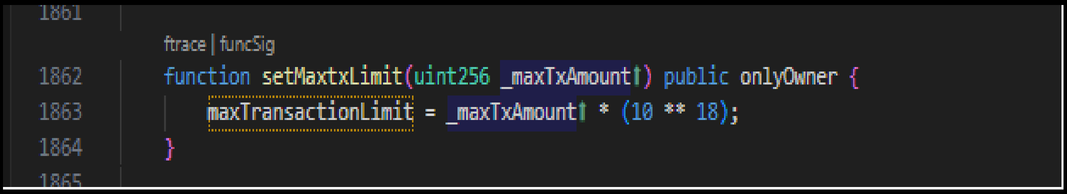


```
ftrace | funcSig
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(_isExcludedFromFees[account] != excluded, "Token: Account is already the value of 'excluded'");
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}

ftrace | funcSig
function setExcludeFromMaxTx(address _address, bool value) public onlyOwner {
    _isExcludedFromLimits[_address] = value;
}
```

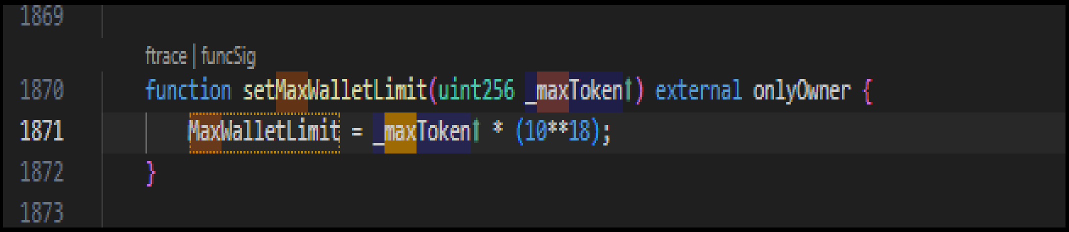
⚠ Max Transaction and Holding Modify function

Contract renounced, function can not be triggered by owner.

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



```
ftrace | funcSig
function setMaxtxLimit(uint256 _maxTxAmount) public onlyOwner {
    maxTransactionLimit = _maxTxAmount * (10 ** 18);
}
```

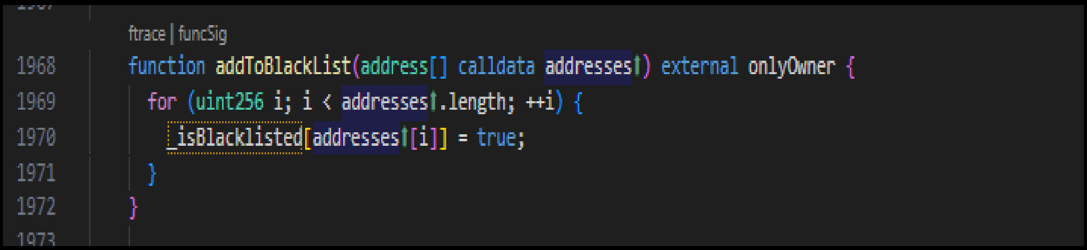


```
ftrace | funcSig
function setMaxWalletLimit(uint256 _maxToken) external onlyOwner {
    MaxWalletLimit = _maxToken * (10**18);
}
```

Blacklist

Contract renounced, function can not be triggered by owner.

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.

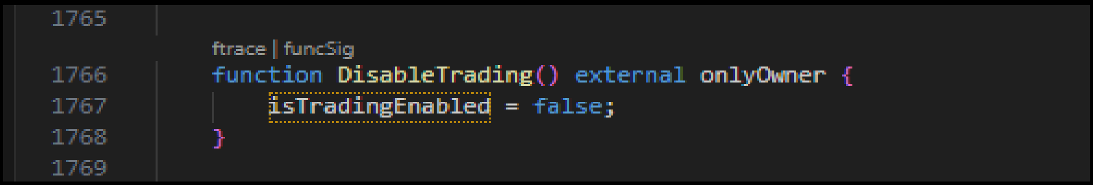


```
ftrace | funcSig
function addToBlackList(address[] calldata addresses) external onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        _isBlacklisted[addresses[i]] = true;
    }
}
```

Trading Disable Function

Contract renounced, function can not be triggered by owner.

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 DisableTrading() external onlyOwner {
    isTradingEnabled = false;
}
```

Contract Security

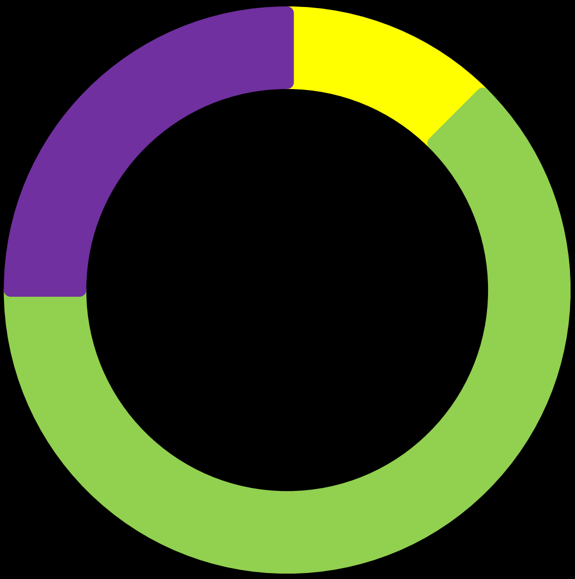
Total Findings: 7

🟥 High 0

🟨 Medium 1

🟩 Low 5

🟪 Info 2



🟥 **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)

- Incorrect Access Control

🟩 Low severity issues: (5)

- Missing Events
- Long number literals
- Outdated Compiler Version
- Upprove Front Running Attack (Sandwich Bot Attack)
- Unchecked Array Lenght

🟪 Informational severity issues: (2)

- Public Functions Should be Declared External
- Missing Zero Address Validation

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

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

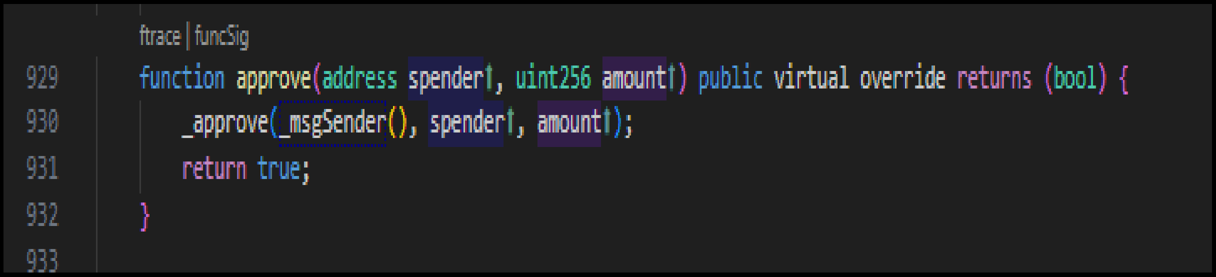
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed	Passed	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed	Passed	Passed
SWC-122	Lack of Proper Signature Verification	Passed	Passed	Passed
SWC-123	Requirement Violation	Passed	Passed	Passed
SWC-124	Write to Arbitrary 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.

⚠ Approve of front running attack. Also known as Sandwich Bot attack. (2 Item)

Item: 1	Location:	Line 929-932	Severity:	🟩 Low

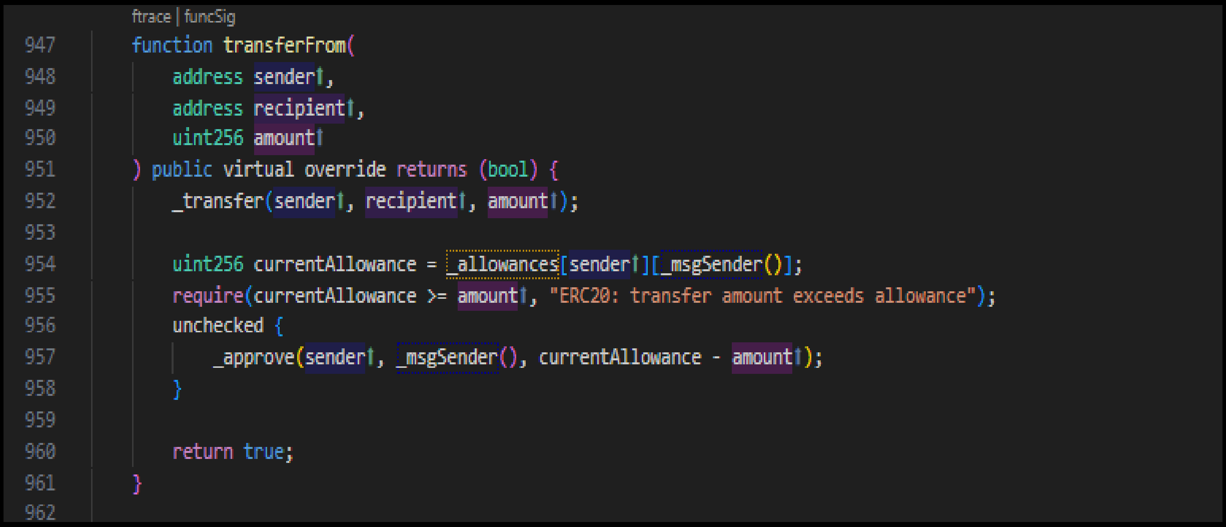
Function	The approve() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver when 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 approve can be front-run by abusing the _approve function.
Remedation	1.Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 2.Use transaction taxes to prevent against front-run attack



```
      ftrace | funcSig
929   function approve(address spender, uint256 amount) public virtual override returns (bool) {
930       _approve(_msgSender(), spender, amount);
931       return true;
932   }
933
```

Item: 2	Location:	Line 947-961	Severity:	🟩 Low

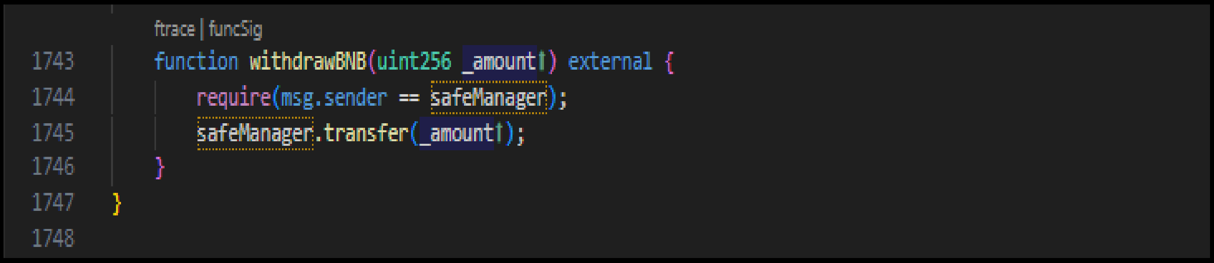
Function	The transferFrom() method overrides current allowance regardless of whether the spender already used it or not, so there is no way to increase or decrease allowance by a certain value atomically unless the token owner is a smart contract, not an account. This can be abused by a token receiver 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 approve can be front-run by abusing the _approve function.
Remedation	1.Introduce mechanisms that limit the maximum acceptable gas price for transactions. This can help prevent front-runners from drastically increasing the gas fees to prioritize their transactions. 2.Use transaction taxes to prevent against front-run attack


```
ftrace | funcSig
function transferFrom(
    address sender,
    address recipient,
    uint256 amount
) public virtual override returns (bool) {
    _transfer(sender, recipient, amount);

    uint256 currentAllowance = _allowances[sender][_msgSender()];
    require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");
    unchecked {
        _approve(sender, _msgSender(), currentAllowance - amount);
    }

    return true;
}
```

⚠ Incorrect Acces Control (1 Item)

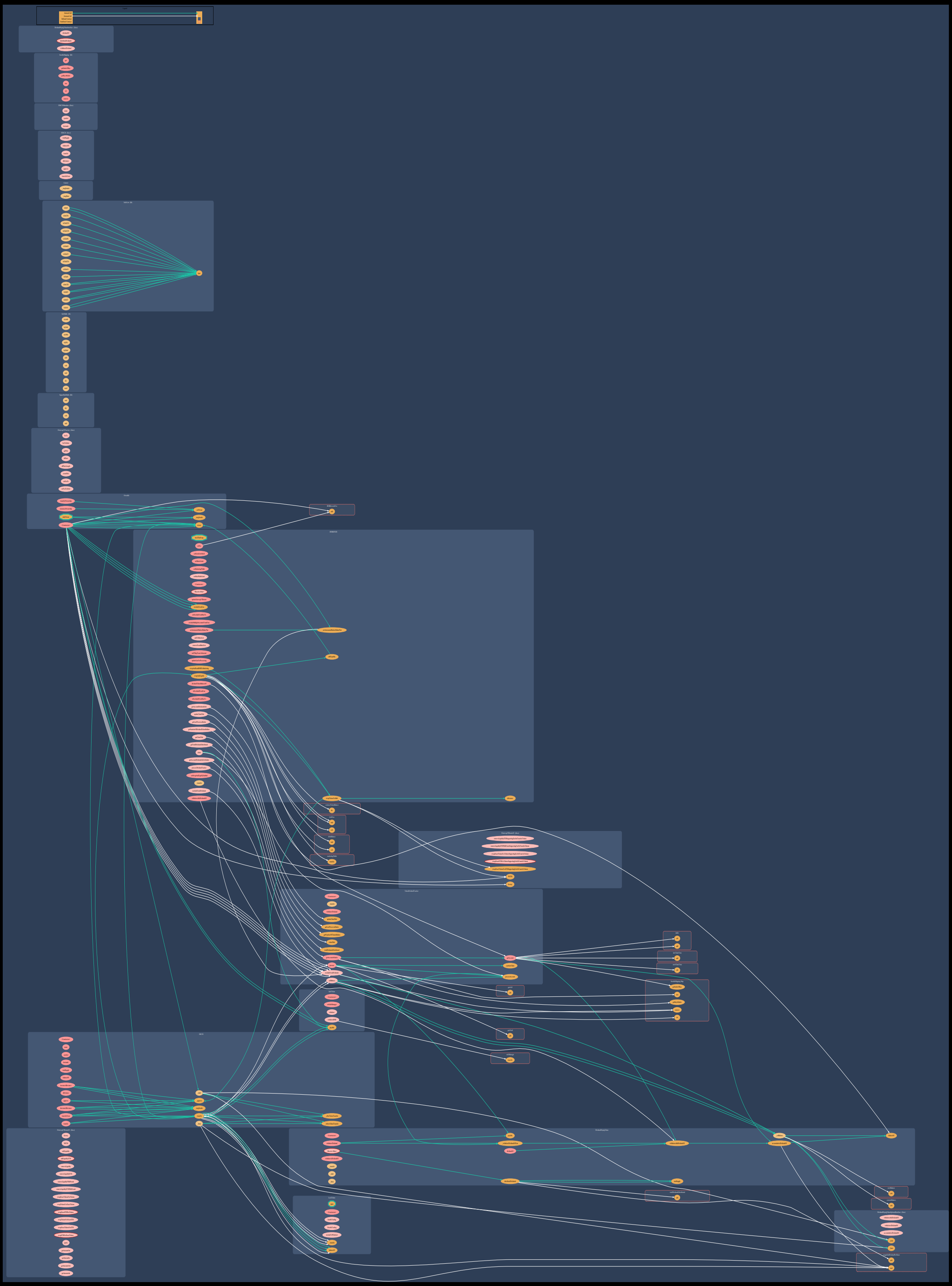
Item: 1	Location:	Line 1743-1746	Severity:	🟨 Medium

Function	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 SafeToken is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function withdrawBNB is missing the modifier onlyOwner.
Remedation	1. Ensure that initialization functions can only be called once and only by authorized entities. 2. Implement least-privilege roles using libraries like OpenZeppelin’s Access Control. 3. Add proper access control modifiers to sensitive functions, such as onlyOwner or custom roles.

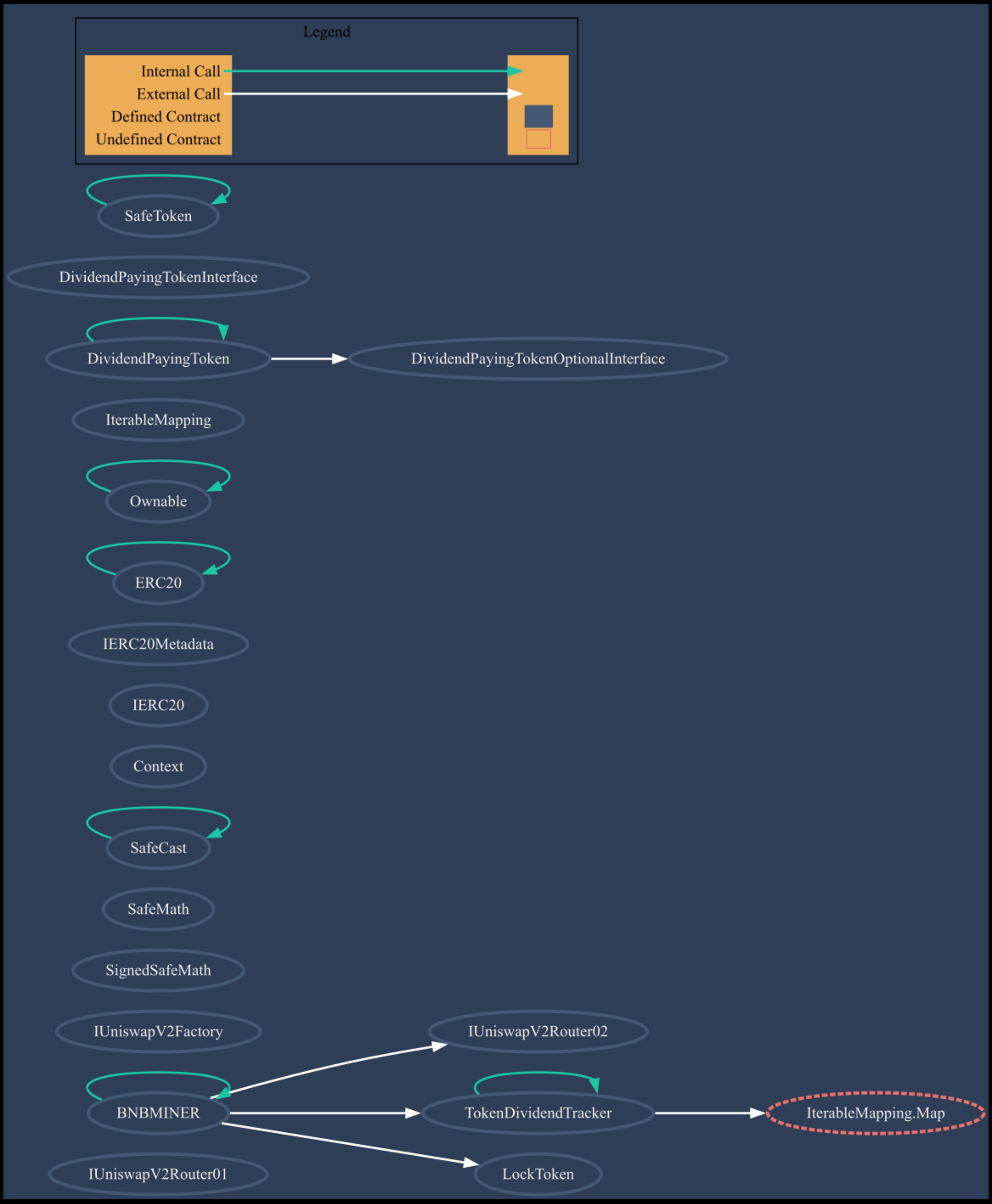


```
    ftrace | funcSig
    function withdrawBNB(uint256 _amount↑) external {
        require(msg.sender == safeManager);
        safeManager.transfer(_amount↑);
    }
}
```

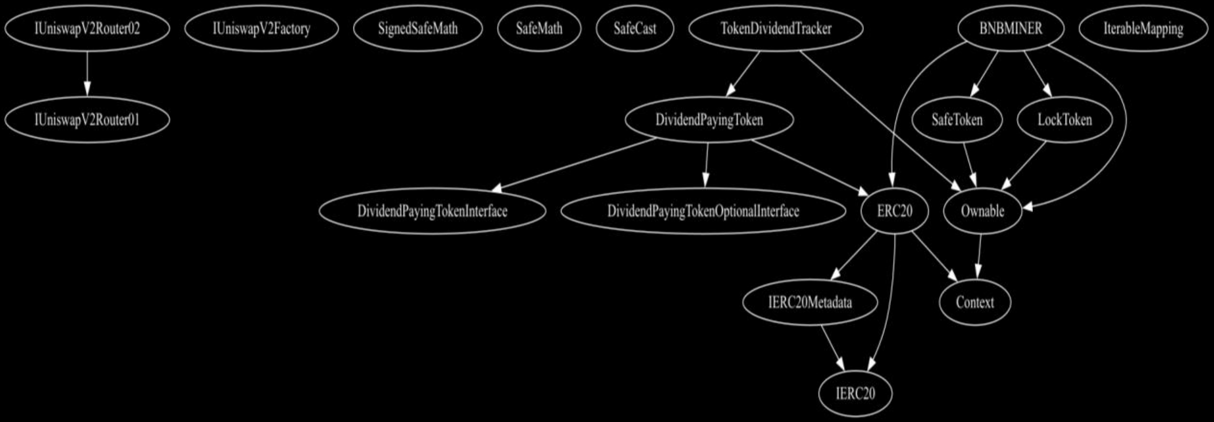
Contract Flow Graph



Contract Interaction Graph



Inheritance Graph



Contract Functions

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
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❗
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❗
SignedSafeMath	Library			
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	sub	Internal 🔒		
└	add	Internal 🔒		
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 🔒		
SafeCast	Library			
└	toUint224	Internal 🔒		
└	toUint128	Internal 🔒		

└	toUint96	Internal 🔒		
└	toUint64	Internal 🔒		
└	toUint32	Internal 🔒		
└	toUint16	Internal 🔒		
└	toUint8	Internal 🔒		
└	toUint256	Internal 🔒		
└	toInt128	Internal 🔒		
└	toInt64	Internal 🔒		
└	toInt32	Internal 🔒		
└	toInt16	Internal 🔒		
└	toInt8	Internal 🔒		
└	toInt256	Internal 🔒		
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❗
IERC20Metadata	Interface	IERC20		

└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
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		
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_setOwner	Private 🔐	🛑	
IterableMapping	Library			
L	get	Public ❗		NO❗
L	getIndexOfKey	Public ❗		NO❗
L	getKeyAtIndex	Public ❗		NO❗
L	size	Public ❗		NO❗
L	set	Public ❗	🛑	NO❗
L	remove	Public ❗	🛑	NO❗
DividendPayingTokenOptionalInterface	Interface			
L	withdrawableDividendOf	External ❗		NO❗
L	withdrawnDividendOf	External ❗		NO❗
L	accumulativeDividendOf	External ❗		NO❗
DividendPayingTokenInterface	Interface			
L	dividendOf	External ❗		NO❗
L	distributeDividends	External ❗	💵	NO❗

L	withdrawDividend	External ❗	🛑	NO❗
DividendPayingToken	Implementation	ERC20, DividendPayingTokenInterface, DividendPayingTokenOptionalInterface		
L		Public ❗	🛑	ERC20
L		External ❗	💵	NO❗
L	distributeDividends	Public ❗	💵	NO❗
L	withdrawDividend	Public ❗	🛑	NO❗
L	_withdrawDividendOfUser	Internal 🔒	🛑	
L	dividendOf	Public ❗		NO❗
L	withdrawableDividendOf	Public ❗		NO❗
L	withdrawnDividendOf	Public ❗		NO❗
L	accumulativeDividendOf	Public ❗		NO❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_setBalance	Internal 🔒	🛑	
TokenDividendTracker	Implementation	DividendPayingToken, Ownable		
L		Public ❗	🛑	DividendPayingToken
L	_transfer	Internal 🔒		

└	withdrawDividend	Public ❗		NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	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
SafeToken	Implementation	Ownable		
└		Public ❗	🛑	NO❗
└	setSafeManager	Public ❗	🛑	onlyOwner
└	withdraw	External ❗	🛑	NO❗
└	withdrawBNB	External ❗	🛑	NO❗
LockToken	Implementation	Ownable		
└		Public ❗	🛑	NO❗
└	EnableTrading	External ❗	🛑	onlyOwner
└	DisableTrading	External ❗	🛑	onlyOwner
└	includeToWhiteList	External ❗	🛑	onlyOwner

BNBMINER	Implementation	ERC20, Ownable, SafeToken, LockToken		
└	setFee	Public ❗	🛑	onlyOwner
└	setExtraFeeOnSell	Public ❗	🛑	onlyOwner
└	setMaxtxLimit	Public ❗	🛑	onlyOwner
└	setMarketingWallet	Public ❗	🛑	onlyOwner
└	setMaxWalletLimit	External ❗	🛑	onlyOwner
└		Public ❗	🛑	ERC20
└		External ❗	💵	NO❗
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	setExcludeFromMaxTx	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	addToBlackList	External ❗	🛑	onlyOwner
└	removeFromBlackList	External ❗	🛑	onlyOwner
└	setSWapToensAtAmount	Public ❗	🛑	onlyOwner
└	updateGasForProcessing	Public ❗	🛑	onlyOwner

L	updateClaimWait	External ❗	🛑	onlyOwner
L	getClaimWait	External ❗		NO❗
L	getTotalDividendsDistributed	External ❗		NO❗
L	isExcludedFromFees	Public ❗		NO❗
L	isExcludedFromMaxTx	Public ❗		NO❗
L	withdrawableDividendOf	Public ❗		NO❗
L	dividendTokenBalanceOf	Public ❗		NO❗
L	getAccountDividendsInfo	External ❗		NO❗
L	getAccountDividendsInfoAtIndex	External ❗		NO❗
L	processDividendTracker	External ❗	🛑	NO❗
L	claim	External ❗	🛑	NO❗
L	getLastProcessedIndex	External ❗		NO❗
L	getNumberOfDividendTokenHolders	External ❗		NO❗
L	excludeFromDividends	External ❗	🛑	onlyOwner
L	setSwapAndLiquifyEnabled	Public ❗	🛑	onlyOwner
L	_transfer	Internal 🔒	🛑	open
L	swapAndLiquify	Private 🔐	🛑	lockTheSwap
L	swapTokensForBnb	Private 🔐	🛑	

L	swapAndSendBNBToMarketing	Private 🔐	🛑	
L	addLiquidity	Private 🔐	🛑	

🛑	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