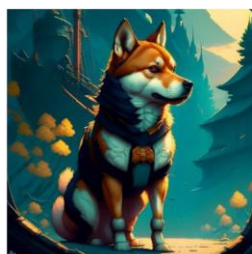


SKELETON ECOSYSTEM

SMART CONTRACT AUDIT



MATSURI SHIBA INU BEP 20

0xBd4B29918D92d613B019252091aB0189f354534f



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	6
Detected Vulnerability Description	10
Contract Flow Chart	18
Inheritance Graph	19
Contract Descriptions	20
Audit Scope	26

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	Matsuri Shiba inu
Ticker/Symbol	MSHIBA
Blockchain	Binance Smart Chain Bep20
Contract Address	0xBd4B29918D92d613B019252091aB0189f354534f
Creator Address	0x6041a5c2E1da3879e12046c764e11A9d3d1C5fEf
Current Owner Address	0x6041a5c2E1da3879e12046c764e11A9d3d1C5fEf
Contract Explorer	https://bscscan.com/token/0xbd4b29918d92d613b019252091ab0189f354534f
Compiler Version	v0.8.17+commit.8df45f5f
License	No License
Optimisation	Yes with 200 Runs
Total Supply	220,000,000,000,000 MSHIBA
Decimals	18

Creation/Audit

Contract Deployed	5 Jun 2023
Audit Created	24-Aug-23 21:00:00 UTC
Audit Update	V 0.1

Verified Socials












Website	https://www.matsurishibainu.com/
Telegram	https://t.me/MatsuriShibaInu
Twitter	https://twitter.com/9yearofshiba













Contract Function Analysis

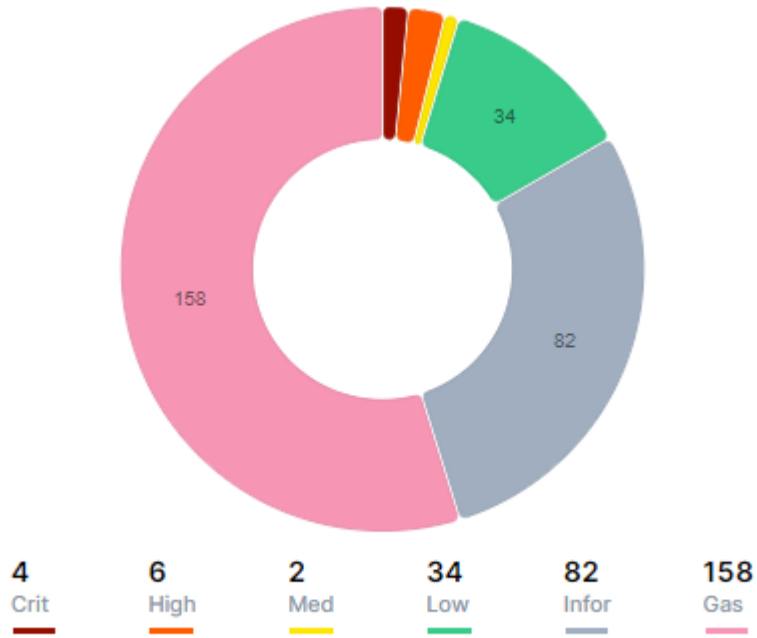
 Pass
  Attention Item
  Risky Item

 Pass
  Attention
  Risk

Contract Verified		The contract source code is uploaded to blockchain explorer and is open source, so everybody can read it.
Contract Renounce		Current Owner: 0x6041a5c2E1da3879e12046c764e11A9d3d1C5fEf Attention marked functions can be modified and used.
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. Setfee max found: 10%
Sell Tax	9%	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: 10%
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		Locked on 22.08.2023: 93% for 647 days on Pinklock Note! Initial liquidity 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. Max Fee Setting option Found! 10% can be set as max 
Proxy Contract		The proxy contract means contract owner can modify 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		<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> <p> If contract is renounced this function still can be used as auto self Destruct</p>
Whitelist Function		<p>Whitelist 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> <p>If there is a whitelist, some addresses may not be able to trade normally (honeypot risk)</p>
Hidden Owner Analysis		<p>No authorised hidden owner found.</p> <p>For contract with a hidden owner, developer can still manipulate the contract even if the ownership has been abandoned. Fake renounce.</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>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.</p> <p>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>Transaction Limiter Function Found.</p> <p>The number of overall token transactions may be limited (honeypot risk)</p>

Contract Safety and Weakness



● INCORRECT ACCESS CONTROL	4
● UNCHECKED TRANSFER	1
● APPROVE FRONT-RUNNING ATTACK	5
● RETURN VALUE OF LOW LEVEL CALLS	1
● DEPRECATED SAFEAPPROVE	1
● USE OWNABLE2STEP	1
● LONG NUMBER LITERALS	1
● USE OF FLOATING PRAGMA	1
● OUTDATED COMPILER VERSION	1
● FUNCTION RETURNS TYPE AND NO RE...	2

● MISSING EVENTS	28
● RETURN INSIDE LOOP	1
● MISSING PAYABLE IN CALL FUNCTION	3
● MISSING STATE VARIABLE VISIBILITY	8
● REQUIRE WITH EMPTY MESSAGE	8
● UNUSED RECEIVE FALLBACK	1
● IN-LINE ASSEMBLY DETECTED	2
● MISSING INDEXED KEYWORDS IN EVE...	1
● USE CALL INSTEAD OF TRANSFER OR ...	3
● BLOCK VALUES AS A PROXY FOR TIME	4


● MISSING UNDERSCORE IN NAMING VA...	51
● USE OF SAFEMATH LIBRARY	2
● CODE OPTIMIZATION BY USING MAX ...	1
● UNNECESSARY CHECKED ARITHMETI...	1
● FUNCTION SHOULD BE EXTERNAL	8
● GAS OPTIMIZATION IN INCREMENTS	1
● SPLITTING REQUIRE STATEMENTS	17
● CHEAPER INEQUALITIES IN IF()	2
● CUSTOM ERRORS TO SAVE GAS	1
● FUNCTION SHOULD RETURN STRUCT	1
● INTERNAL FUNCTIONS NEVER USED	16
● CONSTANT STATE VARIABLES	1
● ARRAY LENGTH CACHING	1
● UNNECESSARY DEFAULT VALUE INITI...	1
● CHEAPER INEQUALITIES IN REQUIRE()	27
● CHEAPER CONDITIONAL OPERATORS	3
● LONG REQUIRE/REVERT STRINGS	9
● VARIABLES DECLARED BUT NEVER US...	14
● STORAGE VARIABLE CACHING IN MEM...	52

 **Incorrect Acces Control (4 item)**

```
1373         if (!_isAllowed[account]) return !_owned[account];
1374         return tokenFromReflection(_owned[account]);
1375     }
1376
1377     function transfer(address recipient, uint256 amount)
1378     public
1379     override
1380     returns (bool)
1381     {
1382         _transfer(_msgSender(), recipient, amount);
1383         return true;
1384     }
1385
1386     function allowance(address owner, address spender)
1387     public
1388     view
1389     override
1390     returns (uint256)
1391     {
1392         return _allowances[owner][spender];
1393     }
1394
1395     function approve(address spender, uint256 amount)
1396     public
1397     override
1398     returns (bool)
1399     {
1400         _approve(_msgSender(), spender, amount);
1401         return true;
1402     }
1403
1404     function transferFrom(
1405         address sender,
1406         address recipient,
1407         uint256 amount
1408     ) public override returns (bool) {
1409         _transfer(sender, recipient, amount);
1410         _approve(
1411             sender,
1412             _msgSender(),
1413             _allowances[sender][_msgSender()].sub(
1414                 amount,
1415                 "ERC20: transfer amount exceeds allowance"
1416             )
1417         );
1418         return true;
1419     }
1420
```

```

1450     function totalFee() public view returns (uint256) {
1451         return _tFeeTotal;
1452     }
1453
1454     function deliver(uint256 tAmount) public {
1455         address sender = _msgSender();
1456         require(
1457             !_isExcluded[sender],
1458             "Excluded addresses cannot call this function"
1459         );
1460         (uint256 rAmount, , , , ) = _getValues(tAmount);
1461         _rOwned[sender] = _rOwned[sender].sub(rAmount);
1462         _rTotal = _rTotal.sub(rAmount);
1463         _tFeeTotal = _tFeeTotal.add(tAmount);
1464     }
1465
1466     function reflectionFromToken(uint256 tAmount, bool deductTransferFee)
1467         public
  
```


Function	Severity	Remediation
<p>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.</p> <p>The contract BABYTOKEN is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function processDividendTracker is missing the modifier onlyOwner.</p>	 Severity : Critical	<p>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</p>


 Unchecked Transfer (1 item)

```

618         uint256 value
619     ) internal {
620         _callOptionalReturn(
621             token,
622             abi.encodeWithSelector(token.transferFrom.selector, from, to, value)
623         );
624     }
625
626     /**
627     * @dev Deprecated. This function has issues similar to the ones found in
628     * {IERC20-approve}, and its usage is discouraged.
629     *

```

Function	Severity	Remediation
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 (5 Item)

```

1386     function allowance(address owner, address spender)
1387         public
1388         view
1389         override
1390         returns (uint256)
1391     {
1392         return _allowances[owner][spender];
1393     }
1394
1395     function approve(address spender, uint256 amount)
1396         public
1397         override
1398         returns (bool)
1399     {
1400         _approve(_msgSender(), spender, amount);
1401         return true;
1402     }
1403

```

```

1404     function transferFrom(
1405         address sender,

```

```

1402     }
1403
1404     function transferFrom(
1405         address sender,
1406         address recipient,
1407         uint256 amount
1408     ) public override returns (bool) {
1409         _transfer(sender, recipient, amount);
1410         _approve(
1411             sender,
1412             _msgSender(),
1413             _allowances[sender][_msgSender()].sub(
1414                 amount,
1415                 "ERC20: transfer amount exceeds allowance"
1416             )
1417         );
1418         return true;
1419     }
1420

```

```

1421     function increaseAllowance(address spender, uint256 addedValue)
1422         public


```

```

1840     }
1841 }
1842
1843     function swapTokensForBNB(uint256 tokenAmount) private {
1844         // generate the uniswap pair path of token -> weth
1845         address[] memory path = new address[](2);
1846         path[0] = address(this);
1847         path[1] = pcsV2Router.WETH();
1848
1849         _approve(address(this), address(pcsV2Router), tokenAmount);
1850
1851         // make the swap
1852         pcsV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
1853             tokenAmount,
1854             0, // accept any amount of ETH
1855             path,
1856             address(this),
1857             block.timestamp
1858         );
1859     }
1860
1861     function swapBNBForTokens(uint256 amount) private {

```

```
1908         IERC20(feeToken).transfer(receiver, newBalance);
1909     }
1910 }
1911
1912     function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
1913         // approve token transfer to cover all possible scenarios
1914         _approve(address(this), address(pcsV2Router), tokenAmount);
1915
1916         address liquidAddr = dead;
1917
1918         if (!burnAutomaticGeneratedLiquidity) {
1919             liquidAddr = owner();
1920         }
1921         // add the liquidity
1922         pcsV2Router.addLiquidityETH{value: ethAmount}(
1923             address(this),
1924             tokenAmount,
1925             0, // slippage is unavoidable
1926             0, // slippage is unavoidable
1927             liquidAddr,
1928             block.timestamp
1929         );
1930     }
1931
1932     //this method is responsible for taking all fee, if takeFee is true
```


Function	Severity	Remediation
<p>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.</p>	 Severity : High	<p>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).</p> <p>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/)</p> <p>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.</p>

⚠ Return value of low level calls (1 Item)

```

564     ) private returns (bytes memory) {
565         require(isContract(target), "Address: call to non-contract");
566
567         // solhint-disable-next-line avoid-low-level-calls
568         (bool success, bytes memory returndata) = target.call{value: weiValue}(
569             data
570         );
571         if (success) {
572             return returndata;
573         } else {
574             // Look for revert reason and bubble it up if present
575             if (returndata.length > 0) {


```

Function	Severity	Remediation
<p>The functions do not check the return value of low-level calls. This can lock Ether in the contract if the call fails or may compromise the contract if the ownership is being changed.</p> <p>The following calls were detected without return value validations - ['call']</p>	 Severity : medium	<p>Ensure return value is checked using conditional statements for low-level calls. We should also ensure that we log failed calls using events.</p>

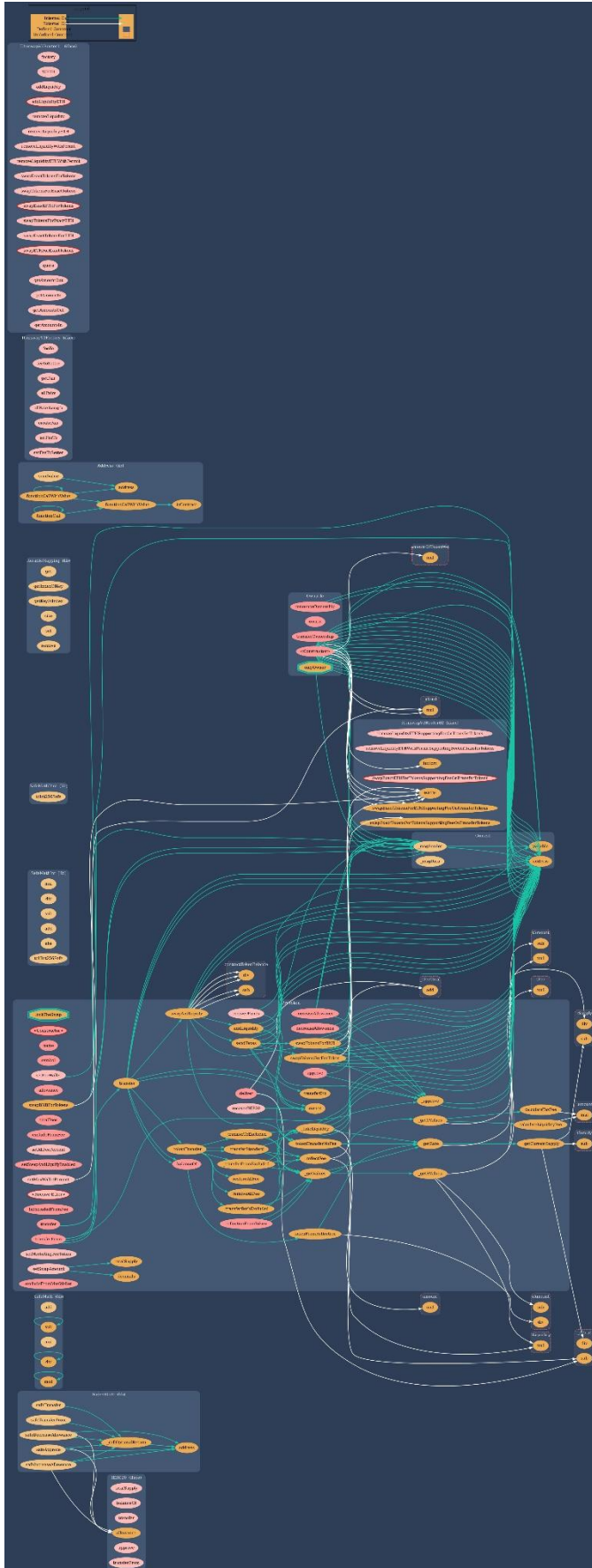
⚠️ Deprecated Safeapprove (1 Item)

```

630     * Whenever possible, use {safeIncreaseAllowance} and
631     * {safeDecreaseAllowance} instead.
632     */
633     function safeApprove(
634         IERC20 token,
635         address spender,
636         uint256 value
637     ) internal {
638         // safeApprove should only be called when setting an initial allowance,
639         // or when resetting it to zero. To increase and decrease it, use
640         // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'
641         // solhint-disable-next-line max-line-length
642         require(
643             (value == 0) || (token.allowance(address(this), spender) == 0),
644             "SafeERC20: approve from non-zero to non-zero allowance"
645         );
646         _callOptionalReturn(
647             token,
648             abi.encodeWithSelector(token.approve.selector, spender, value)
649         );
650     }
651
652     function safeIncreaseAllowance(
653         IERC20 token,
  
```

Function	Severity	Remediation
OpenZeppelin's <code>safeApprove()</code> function is deprecated and should not be used since it is affected by similar issues as <code>approve()</code> and can be exploited in frontrunning or sandwich attacks.	 Severity : medium	It is recommended to use <code>safeIncreaseAllowance()</code> and <code>safeDecreaseAllowance</code> instead of <code>safeApprove()</code> .





















Contract Flow Graph






























Inheritance Graph




























Contract Descriptions

























Contract	Type	Bases		
		Visibility	Mutability	Modifiers
	Function Name			
IERC20	Interface			
	totalSupply	External !		NO!
	balanceOf	External !		NO!
	transfer	External !		NO!
	allowance	External !		NO!
	approve	External !		NO!
	transferFrom	External !		NO!
SafeMath	Library			
	add	Internal 		
	sub	Internal 		
	sub	Internal 		
	mul	Internal 		
	div	Internal 		
	div	Internal 		
	mod	Internal 		
	mod	Internal 		
Context	Implementation			
	_msgSender	Internal 		
	_msgData	Internal 		
SafeMathInt	Library			
	mul	Internal 		
	div	Internal 		
	sub	Internal 		
	add	Internal 		
	abs	Internal 		
	toUint256Safe	Internal 		
SafeMathUint	Library			
	toInt256Safe	Internal 		

IterableMapping	Library			
	get	Internal 		
	getIndexOfKey	Internal 		
	getKeyAtIndex	Internal 		
	size	Internal 		
	set	Internal 		
	remove	Internal 		
Address	Library			
	isContract	Internal 		
	sendValue	Internal 		
	functionCall	Internal 		
	functionCall	Internal 		
	functionCallWithValue	Internal 		
	functionCallWithValue	Internal 		
	_functionCallWithValue	Private 		
SafeERC20	Library			
	safeTransfer	Internal 		
	safeTransferFrom	Internal 		
	safeApprove	Internal 		
	safeIncreaseAllowance	Internal 		
	safeDecreaseAllowance	Internal 		
	_callOptionalReturn	Private 		
Ownable	Implementation	Context		
		Public 		NO 
	owner	Public 		NO 
	renounceOwnership	Public 		onlyOwner
	transferOwnership	Public 		onlyOwner

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!
ProToken	Implementation	Context, IERC20, Ownable		
		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!
	totalFees	Public !		NO!

	deliver	Public !		NO!
	reflectionFromToken	Public !		NO!
	tokenFromReflection	Public !		NO!
	excludeFromFee	Public !		onlyOwner
	setAllFeePercent	External !		onlyOwner
	setSwapAndLiquifyEnabled	Public !		onlyOwner
	setSwapAmount	External !		onlyOwner
		External !		NO!
	_reflectFee	Private 		
	_getValues	Private 		
	_getTValues	Private 		
	_getRValues	Private 		
	_getRate	Private 		
	_getCurrentSupply	Private 		
	_takeLiquidity	Private 		
	calculateTaxFee	Private 		
	calculateLiquidityFee	Private 		
	removeAllFee	Private 		
	restoreAllFee	Private 		
	isExcludedFromFee	Public !		NO!
	_approve	Private 		
	_transfer	Private 		
	swapAndLiquify	Private 		lockThe Swap
	swapTokensForBNB	Private 		
	swapBNBForTokens	Private 		
	swapTokensForFeeToken	Private 		
	addLiquidity	Private 		
	_tokenTransfer	Private 		
	_transferStandard	Private 		

	_transferToExcluded	Private 		
	_transferFromExcluded	Private 		
	_transferBothExcluded	Private 		
	_tokenTransferNoFee	Private 		
	transferEth	Private 		
	recoverFunds	External 		onlyOwner
	recoverBEP20	External 		onlyOwner
	sendTaxes	Internal 		
	setFeeWallet	External 		onlyOwner
	setMarketingFeeToken	External 		onlyOwner
	setMaxWalletPercent	External 		onlyOwner
	excludeFromMaxWallet	Public 		onlyOwner



Function
can modify
state



Function
is payable

Source:

File Name SHA-1 Hash

c:\Solidity\maturishiba.sol

59466a587e8ee5dd6882dbd8146752350f19b513

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>

