SecureWise

AUDIT REPORT

SecureWise

BABY WIKICAT

SecureWise

Table of Contents

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

Overview

Token Name: Baby WikiCat **(BWKC)**

Methodology: Automated Analysis, Manual Code Review

Language: Solidity

Contract Address: 0x99e4ffeE5367592A82B5d4Be6F30144F400388fC

ContractLink: https://bscscan.com/address/0x99e4ffeE5367592A82B5d4Be6F30144F400388fC

Network: Binance Smart Chain (BSC)

Decimals: 9

Supply: 1.000.000.000.000.000

Website: https://www.babywikicat.co

Twitter: https://twitter.com/Babywkc

Telegram: https://t.me/Babywkc

Report Date: February 21, 2023

Quick Result

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

	The owner can pause trading
	The owner can set fees up to 100%
	The owner can set the max tx amount "0"
	Auto liquidity is going to an externally owned account
	The owner can exclude accounts from rewards
	The owner can set a blacklist and can use it to block any account from trading.
	The owner can change transaction lock-time without limit
	The owner can change max wallet token amount to "0"
	The owner can exclude accounts from fees
	The owner can exclude accounts from transaction limit
	The owner can exclude accounts from time lock limit
	The owner can change swap settings
	The owner can withdraw any token from the contract

visit **page 13** for all details

Baby WikiCat (BWKC) has succesfully **PASSED** the smart contract audit with **HIGH** , **MEDIUM** and **LOW** severity issue

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fxed as soon as possible.

Low: Effects are minimal in isolation and do not pose a signifcant danger to the project or its users. Issues under this classifcation are recommended to be fixed nonetheless.

Automated Analysis

Symbol	Meaning
🔴	Function can modify state
💵	Function is payable

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
Project	Implementation			
SafeMath	Library			
L	add	Internal 🔒		
L	sub	Internal 🔒		
L	sub	Internal 🔒		
L	mul	Internal 🔒		
L	div	Internal 🔒		
L	div	Internal 🔒		
IBEP20	Interface			
L	totalSupply	External ❗		NO ❗
L	decimals	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	name	External ❗		NO ❗
L	getOwner	External ❗		NO ❗
L	balanceOf	External ❗		NO ❗
L	transfer	External ❗	🔴	NO ❗
L	allowance	External ❗		NO ❗
L	approve	External ❗	🔴	NO ❗
L	transferFrom	External ❗	🔴	NO ❗
Auth	Implementation			
L		Public ❗	🔴	NO ❗
L	authorize	Public ❗	🔴	onlyOwner
L	unauthorize	Public ❗	🔴	onlyOwner
L	isOwner	Public ❗		NO ❗
L	isAuthorized	Public ❗		NO ❗

Automated Analysis

└	transferOwnership	Public !	🔴	onlyOwner
IDEXFactory	Interface			
└	createPair	External !	🔴	NO !
IDEXRouter	Interface			
└	factory	External !		NO !
└	WETH	External !		NO !
└	addLiquidity	External !	🔴	NO !
└	addLiquidityETH	External !	💵	NO !
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External !	🔴	NO !
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External !	💵	NO !
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External !	🔴	NO !
IDividendDistributor	Interface			
└	setDistributionCriteria	External !	🔴	NO !
└	setShare	External !	🔴	NO !
└	deposit	External !	💵	NO !
└	process	External !	🔴	NO !
DividendDistributor	Implementation	IDividendDistributor		
└		Public !	🔴	NO !
└	setDistributionCriteria	External !	🔴	onlyToken
└	setShare	External !	🔴	onlyToken
└	deposit	External !	💵	onlyToken
└	process	External !	🔴	onlyToken
└	shouldDistribute	Internal 🔒		
└	distributeDividend	Internal 🔒	🔴	
└	claimDividend	External !	🔴	NO !
└	getUnpaidEarnings	Public !		NO !

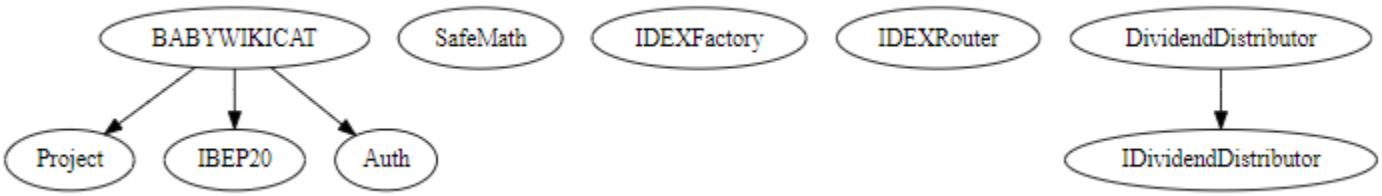
Automated Analysis

└	getCumulativeDividends	Internal 🔒		
└	addShareholder	Internal 🔒	🔴	
└	removeShareholder	Internal 🔒	🔴	
BABYWIKICAT	Implementation	Project, IBEP20, Auth		
└		Public ❗	🔴	Auth
└		External ❗	💵	NO ❗
└	totalSupply	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	name	External ❗		NO ❗
└	getOwner	External ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	Public ❗	🔴	NO ❗
└	approveMax	External ❗	🔴	NO ❗
└	transfer	External ❗	🔴	NO ❗
└	transferFrom	External ❗	🔴	NO ❗
└	setMaxWalletPercent_base1000	External ❗	🔴	onlyOwner
└	setMaxTxPercent_base1000	External ❗	🔴	onlyOwner
└	setBuyTax	External ❗	🔴	onlyOwner
└	setTxLimit	External ❗	🔴	authorized
└	setBurnTo	External ❗	🔴	onlyOwner
└	setBuyBurnFee	External ❗	🔴	onlyOwner
└	setSwapBurnFee	External ❗	🔴	onlyOwner
└	_transferFrom	Internal 🔒	🔴	
└	_basicTransfer	Internal 🔒	🔴	
└	checkTxLimit	Internal 🔒		
└	shouldTakeFee	Internal 🔒		
└	takeFee	Internal 🔒	🔴	

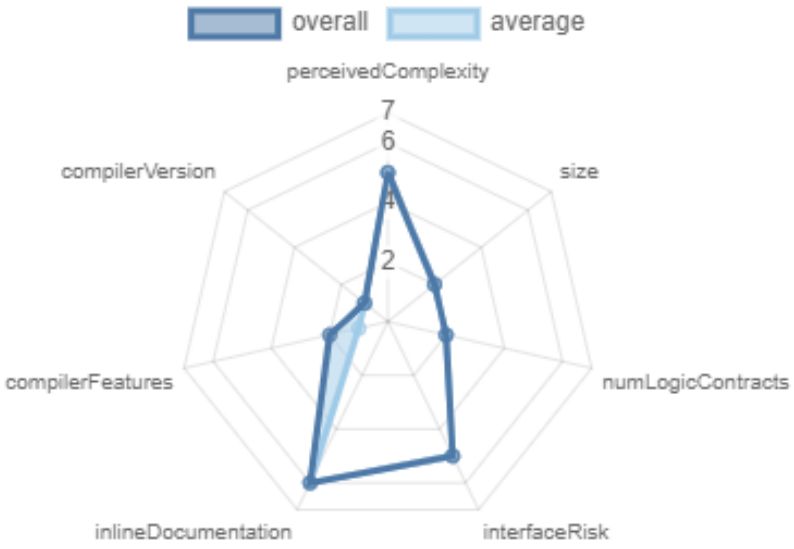
Automated Analysis

L	shouldSwapBack	Internal 🔒		
L	clearStuckBalance	External !	●	authorized
L	clearStuckBalance_sender	External !	●	authorized
L	tradingStatus	Public !	●	onlyOwner
L	cooldownEnabled	Public !	●	onlyOwner
L	swapBack	Internal 🔒	●	swapping
L	setIsDividendExempt	External !	●	authorized
L	manage*blacklist*and*dividend*exempt	Public !	●	onlyOwner
L	manage*burn*exempt	Public !	●	onlyOwner
L	enable_blacklist	Public !	●	onlyOwner
L	manage_blacklist	Public !	●	onlyOwner
L	setIsFeeExempt	External !	●	authorized
L	setIsTxLimitExempt	External !	●	authorized
L	setIsTimelockExempt	External !	●	authorized
L	setSwapFees	External !	●	authorized
L	setTreasuryFeeReceiver	External !	●	authorized
L	setMarketingWallet	External !	●	authorized
L	setFeeReceivers	External !	●	authorized
L	setSwapBackSettings	External !	●	authorized
L	setTargetLiquidity	External !	●	authorized
L	setDistributionCriteria	External !	●	authorized
L	setDistributorSettings	External !	●	authorized
L	getCirculatingSupply	Public !		NO !
L	getLiquidityBacking	Public !		NO !
L	isOverLiquified	Public !		NO !
L	multiTransfer	External !	●	onlyOwner
L	multiTransfer_fixed	External !	●	onlyOwner

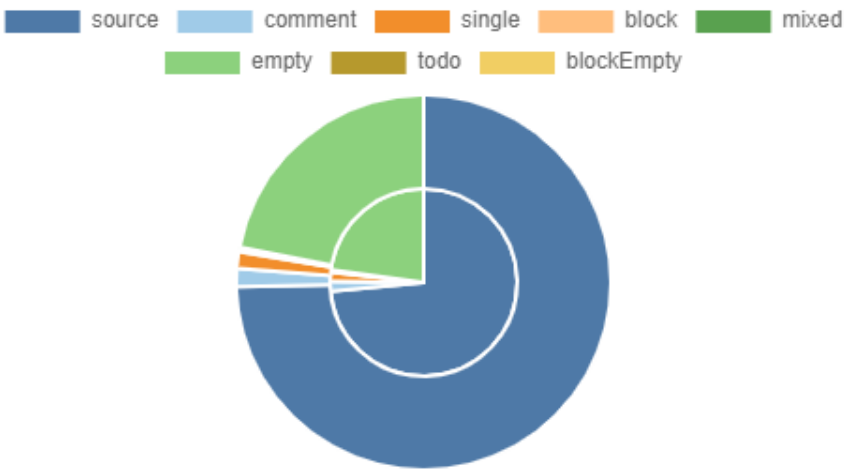
Inheritance Graph



Risk



Source Lines



Contract Summary

Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
5	4	815	745	560	13	579	
5	4	815	745	560	13	579	

Components

Contracts	Libraries	Interfaces	Abstract
2	1	4	2

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Public	Payable
77	5

External	Internal	Private	Pure	View
60	78	0	11	22

StateVariables

Total	Public
63	33

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
0.7.6		yes		

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
yes					yes → NewContract:DividendDistributor

TryCatch	Σ Unchecked
yes	

Manual Review

The owner can pause trading

```
586     // switch Trading
        0 references | Control flow graph | 0d295980
587     function tradingStatus(bool _status) public onlyOwner {
588         tradingOpen = _status;
589     }
```

Privileged roles can be granted the power to pause() the contract in case of an external attack Remove pause trading function.

Recommendation

These functions should be removed this Major problem. Cause can affect decentralization.

The owner can set the max tx amount "0"

```
        0 references | Control flow graph | 5c85974f
466     function setTxLimit(uint256 amount) external authorized {
467         _maxTxAmount = amount;
468     }
469
```

```
        0 references | Control flow graph | bd9ab537
458     function setMaxTxPercent_base1000(uint256 maxTXPercentage_base1000) external onlyOwner() {
459         _maxTxAmount = (_totalSupply * maxTXPercentage_base1000 ) / 1000;
460     }
461
```

These functions are set without any arbitrary limits.

Recommendation

should be provide arbitrary limits. e.g., put a require check that allows minimum /maximum limit etc. if set 0 these cause pause the trading.

The owner can set fees up to 100%

`setSwapBurnFee()`
`setBuyBurnFee()`
`setBuyTax()`

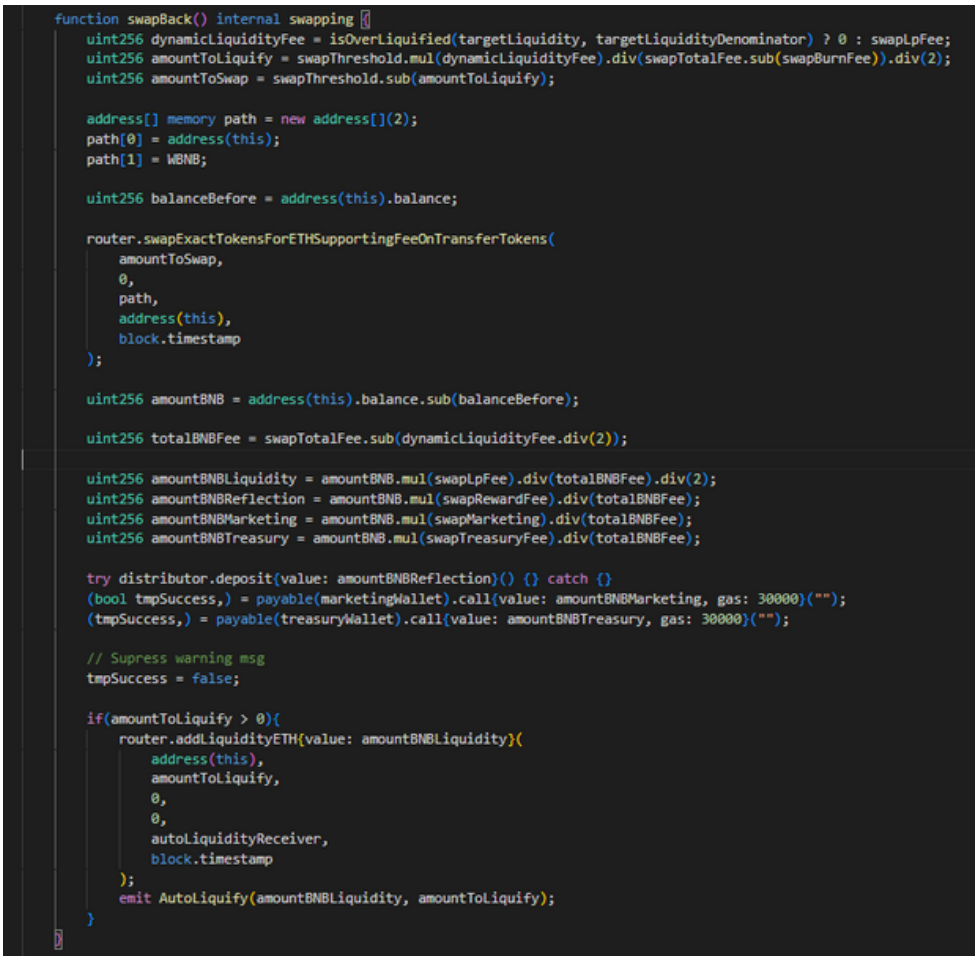
These functions are set without any arbitrary limits.

Recommendation

Should be provide arbitrary limits. e.g., put a require check that allows minimum /maximum limit etc.

Manual Review

Auto liquidity is going to an externally owned account



```
function swapBack() internal swapping {
    uint256 dynamicLiquidityFee = isOverLiquified(targetLiquidity, targetLiquidityDenominator) ? 0 : swapLpFee;
    uint256 amountToLiquify = swapThreshold.mul(dynamicLiquidityFee).div(swapTotalFee.sub(swapBurnFee)).div(2);
    uint256 amountToSwap = swapThreshold.sub(amountToLiquify);

    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = WBNB;

    uint256 balanceBefore = address(this).balance;

    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        amountToSwap,
        0,
        path,
        address(this),
        block.timestamp
    );

    uint256 amountBNB = address(this).balance.sub(balanceBefore);

    uint256 totalBNBFee = swapTotalFee.sub(dynamicLiquidityFee.div(2));

    uint256 amountBNBLiquidity = amountBNB.mul(swapLpFee).div(totalBNBFee).div(2);
    uint256 amountBNBReflection = amountBNB.mul(swapRewardFee).div(totalBNBFee);
    uint256 amountBNBMarketing = amountBNB.mul(swapMarketing).div(totalBNBFee);
    uint256 amountBNBTreasury = amountBNB.mul(swapTreasuryFee).div(totalBNBFee);

    try distributor.deposit{value: amountBNBReflection}() {} catch {}
    (bool tmpSuccess,) = payable(marketingWallet).call{value: amountBNBMarketing, gas: 30000}("");
    (tmpSuccess,) = payable(treasuryWallet).call{value: amountBNBTreasury, gas: 30000}("");

    // Supress warning msg
    tmpSuccess = false;

    if(amountToLiquify > 0){
        router.addLiquidityETH{value: amountBNBLiquidity}(
            address(this),
            amountToLiquify,
            0,
            0,
            autoLiquidityReceiver,
            block.timestamp
        );
        emit AutoLiquify(amountBNBLiquidity, amountToLiquify);
    }
}
```

Authorizing privileged roles to externally-owned-account (EOA) is dangerous.

Recommendation

Send LP tokens to dead address or unreachable address.

Manual Review

The owner can exclude accounts from rewards

```
646    function setIsDividendExempt(address holder, bool exempt) external authorized {
647        require(holder != address(this) && holder != pair);
648        isDividendExempt[holder] = exempt;
649        if(exempt){
650            distributor.setShare(holder, 0);
651        }else{
652            distributor.setShare(holder, _balances[holder]);
653        }
654    }
```

Authorizing privileged roles to exclude accounts from rewards.

Recommendation

These function cause can affect decentralization.

The owner can set a blacklist and can use it to block any account from trading.

```
656    function manage_blacklist_and_dividend_exempt(address[] calldata addresses, bool exempt) public onlyOwner {
657        for (uint256 i; i < addresses.length; ++i) {
658            address holder = addresses[i];
659            require(holder != address(this) && holder != pair);
660            isDividendExempt[holder] = exempt;
661            if(exempt){
662                distributor.setShare(holder, 0);
663            } else{
664                distributor.setShare(holder, _balances[holder]);
665            }
666
667            isBlacklisted[holder] = exempt;
668        }
669    }
```

```
681    function manage_blacklist(address[] calldata addresses, bool status) public onlyOwner {
682        for (uint256 i; i < addresses.length; ++i) {
683            isBlacklisted[addresses[i]] = status;
684        }
685    }
```

Authorizing privileged roles to add an account to black list and pause trade for any account.

Recommendation

These function cause can affect decentralization. Remove Blacklist.

Manual Review

The owner can change transaction lock-time without limit

```
591     // enable cooldown between trades
592     function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
593         buyCooldownEnabled = _status;
594         cooldownTimerInterval = _interval;
595     }
```

Authorizing privileged roles to change transaction lock-time without limit and these cause by semi pause trading.

Recommendation

Put a require check that allows maximum limit etc.

The owner can change max wallet token amount to "0"

```
455     function setMaxWalletPercent_base1000(uint256 maxWallPercent_base1000) external onlyOwner() {
456         _maxWalletToken = (_totalSupply * maxWallPercent_base1000 ) / 1000;
457     }
```

Authorizing privileged roles to change max wallet amount without limit and these cause by semi pause trading. shouldn't be 0.

Recommendation

Put a require check that allows minimum arbitrary limit etc.

The owner can exclude accounts from fees

```
688     function setIsFeeExempt(address holder, bool exempt) external authorized {
689         isFeeExempt[holder] = exempt;
690     }
```

Authorizing privileged roles to exclude accounts from fees.

Recommendation

These function cause can affect decentralization.

Manual Review

The owner can exclude accounts from TX limit, time lock limit, burn exempt

`manage_burn_exempt()`
`setIsTxLimitExempt()`
`setIsTimelockExempt()`

These functions are set without any arbitrary limits.

Recommendation

Should be provide arbitrary limits. e.g., put a require check that allows minimum /maximum limit etc.

The owner can change swap settings

```
736    function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
737        swapEnabled = _enabled;
738        swapThreshold = _amount;
739    }
740
```

The owner can set new swap settings. this can affect functionality of smart contract.

Recommendation

These function cause can affect decentralization

The owner can withdraw any token from the contract

```
576    function clearStuckBalance(uint256 amountPercentage) external authorized {
577        uint256 amountBNB = address(this).balance;
578        payable(marketingWallet).transfer(amountBNB * amountPercentage / 100);
579    }
580
581    function clearStuckBalance_sender(uint256 amountPercentage) external authorized {
582        uint256 amountBNB = address(this).balance;
583        payable(msg.sender).transfer(amountBNB * amountPercentage / 100);
584    }
```

The owner can withdraw any token from the contract. and can be withdrawn by the authorized person too.

Recommendation

These function cause can affect decentralization. Remove or add reqiure for native token. only withdraw stuck token. not native token.

Manual Review

Unchecked return value

```
307     function distributeDividend(address shareholder) internal {
308         if(shares[shareholder].amount == 0){ return; }
309
310         uint256 amount = getUnpaidEarnings(shareholder);
311         if(amount > 0){
312             totalDistributed = totalDistributed.add(amount);
313             RWRD.transfer(shareholder, amount);
314             shareholderClaims[shareholder] = block.timestamp;
315             shares[shareholder].totalRealised = shares[shareholder].totalRealised.add(amount);
316             shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount);
317         }
318     }
```

```
/* Airdrop */
function multiTransfer(address from, address[] calldata addresses, uint256[] calldata tokens) external onlyOwner {

    require(addresses.length < 501,"GAS Error: max airdrop limit is 500 addresses");
    require(addresses.length == tokens.length,"Mismatch between Address and token count");

    uint256 SCCC = 0;

    for(uint i=0; i < addresses.length; i++){
        SCCC = SCCC + tokens[i];
    }

    require(balanceOf(from) >= SCCC, "Not enough tokens in wallet");

    for(uint i=0; i < addresses.length; i++){
        _basicTransfer(from,addresses[i],tokens[i]);
        if(!isDividendExempt[addresses[i]]) {
            try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
        }
    }
```

```
795   function multiTransfer_fixed(address from, address[] calldata addresses, uint256 tokens) external onlyOwner {
796
797       require(addresses.length < 801,"GAS Error: max airdrop limit is 800 addresses");
798
799       uint256 SCCC = tokens * addresses.length;
800
801       require(balanceOf(from) >= SCCC, "Not enough tokens in wallet");
802
803       for(uint i=0; i < addresses.length; i++){
804           _basicTransfer(from,addresses[i],tokens);
805           if(!isDividendExempt[addresses[i]]) {
806               try distributor.setShare(addresses[i], _balances[addresses[i]]) {} catch {}
807           }
808       }
809
810       // Dividend tracker
811       if(!isDividendExempt[from]) {
812           try distributor.setShare(from, _balances[from]) {} catch {}
813       }
814   }
```

If the return value of a low-level call is not checked, the execution may resume even if the function call throws an error. This can lead to unexpected behaviour and break the program logic. A failed call can even be caused by an attacker, who may be able to further exploit the contract.

Recommendation

In the case that you use low-level calls, be sure to check the return value to handle possible failed calls.

Manual Review

Lacks a zero-check on set wallets function

```
setTreasuryFeeReceiver()
setMarketingWallet()
setFeeReceivers()
setBurnTo()
```

Zero-address checks as input validation on address parameters is always a best practice. This is especially true for critical addresses that are immutable and set in the constructor because they cannot be changed later. Accidentally using zero addresses here will lead to failing logic or force contract redeployment and increased gas costs.

Recommendation

Add zero-address input validation for these addresses.

Burn address shouldn't constant and dead wallet

```
470     function setBurnTo(address newBurnTo) external onlyOwner() {
471         burnTo = newBurnTo;
472     }
```

The owner can burn token but burn wallet can be change The owner can set any address.

Recommendation

These function cause can affect decentralization. Remove update burn address etc. should be declared constand dead zero wallet.

Manual Review

Access Modifiers Vulnerabilities

```
manage_blacklist()
manage_burn_exempt()
transferOwnership()
enable_blacklist()
cooldownEnabled()
authorize()
tradingStatus()
manage_blacklist_and_dividend_exempt()
```

These functions are used as public instead of external.

Recommendation

Access control identifiers must be authenticated and set adequately to avoid possible vulnerabilities

Out date compiler version

```
pragma solidity 0.7.6;
```

Compiler is set an outdated version.

Recommendation

Set Compiler to version 0.8.18

SecureWise

AUDIT REPORT

SecureWise

https://securewise.info/

https://t.me/securewisehub

https://github.com/securewise