CheckPoint

Token Security Audit Report Prepared for Baby White Hamster

[v.1.0]

September 2021

Document Properties

Client	Baby White Hamster
Platform	Binance Smart Chain
Language	Solidity
Codebase	0x4ed2bBfc2fC7b0B7400D8D12248649045e82c708

Audit Summary

Delivery Date	06.09.2021
Audit Methodology	Static Analysis, Manual Review
Auditor(s)	Erno Patiala
Classification	Publlic
Version	1.0

Contact Information

Company	CheckPoint
Name	Hanna Järvinen
Telegram	t.me/checkpointreport
E-mail	contact@checkpoint.report

Remark: For more information about this document and its contents, please contact CheckPoint team

Table Of Contents

1 Executive Summary

On 06/09/2021, CheckPoint conducted a full audit for the Baby White Hamster to verify the overall security posture including a smart contract review to discover issues and vulnerabilities in the source code. Static Code Analysis, Dynamic Analysis, and Manual Review werdone in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem.

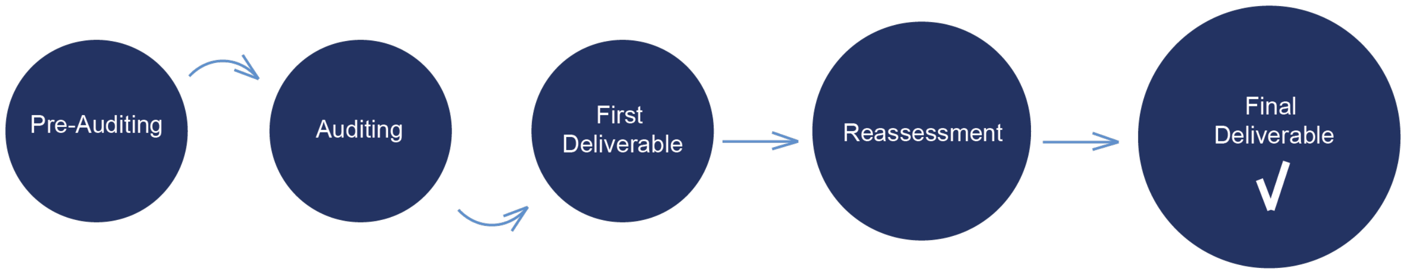
After further analysis and internal discussion, we determined a few issues of varying severities that need to be brought up and paid more attention to. More information can be found in **Section 5 'Audit Result'**. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

🟩 **Baby White Hamster**
Low Risk Level

Communication Channels	Website Content Analysis, Social Media Listening
Smart Contract Code	Smart Contract Details, Contract Function Details, Issues Checking Status, Detailed Findings Information

THIS TOKEN PASSES CHECKPOINT'S SECURITY VERIFICATION STANDART

2 Audit Methodology



CheckPoint conducts the following procedure to enhance the security level of our clients' tokens:

- **Pre-Auditing**

  Planning a comprehensive survey of the token, its ecosystem, possible risks & prospects, getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing.

- **Auditing**

  Study of all available information about the token on the Web, inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals.

- **First Deliverable and Consulting**

  Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation.

- **Reassessment**

  Verifying the status of the issues and whether there are any other complications in the fixes applied.

- **Final Deliverable**

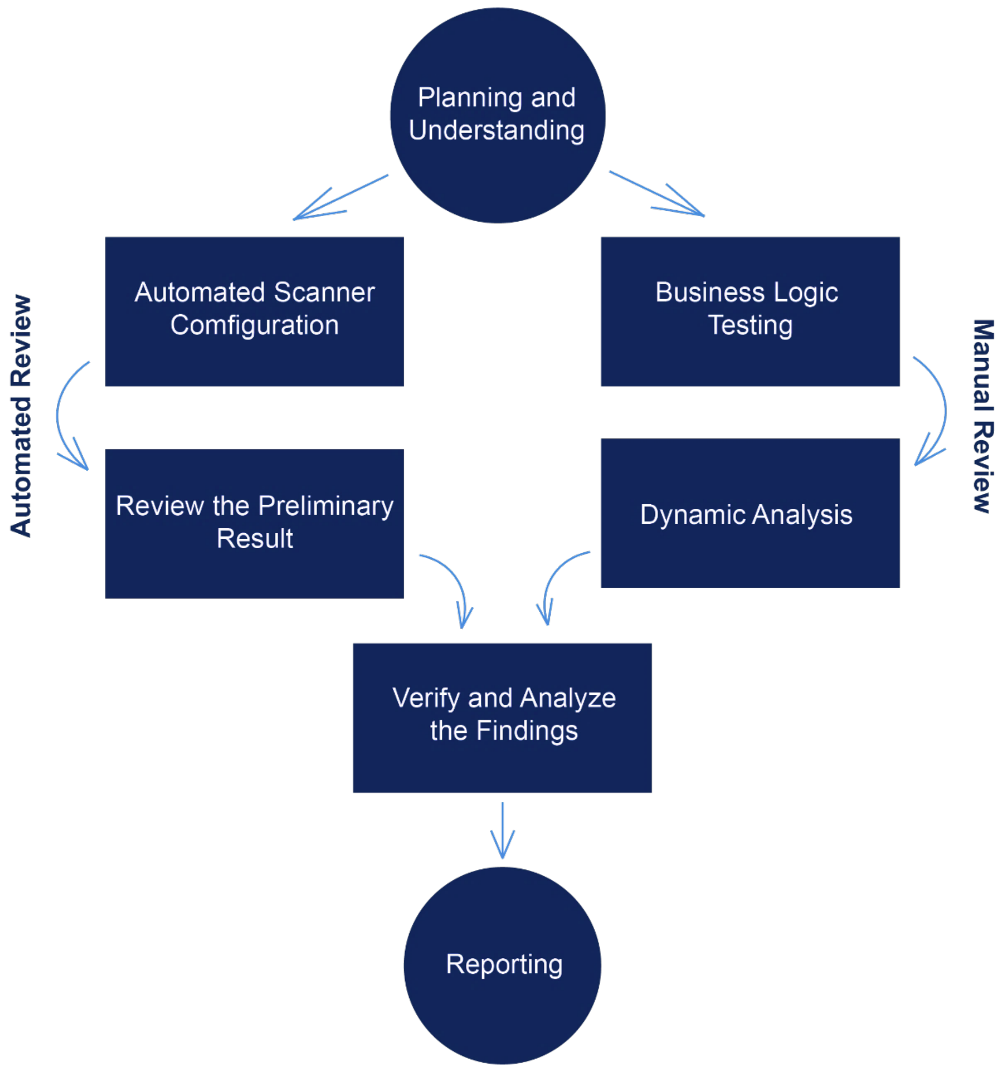
  Providing a full report with the detailed status of each issue.

The security audit process of CheckPoint includes three types testing:

1. Examining publicly available information about the token on social networks, including a detailed overview of the official website and analysis of the latest messages and opinions about the token.

2. Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

3. Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.


Planning and Understanding
Automated Scanner Comfiguration
Business Logic Testing
Automated Review
Manual Review
Review the Preliminary Result
Dynamic Analysis
Verify and Analyze the Findings
Reporting

Remark: Manual and Automated review approaches can be mixed and matched including business logic analysis in terms of malicious doers' perspective

In particular, we perform the audit according to the following procedure:

- **Planning & Understanding**
    - determine scope of testing and understand application purpose and workflows;
    - identify key risk areas, including technical and business risks;
    - determine approach – which sections to review within the resource constraints and review method – automated, manual or mixed.
- **Automated Review**
    - adjust automated source code review tools to inspect the code for known unsafe coding patterns;
    - verify output of the tool in order to eliminate false positive result, and if necessary, adjust and re-run the code review tool.
- **Manual Review**
    - testing for business logic flaws requires thinking in unconventional methods;
    - identify unsafe coding behavior via static code analysis.
- **Reporting**
    - analyze the root cause of the flaws;
    - recommend coding process improvements.

3 Risk Level Classification

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

- **Likelihood** represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- **Impact** measures the technical loss and business damage of a successful attack.
- **Severity** demonstrates the overall criticality of the risk and calculated as the product of impact and likelihood values, illustrated in a twodimensional matrix. The shading of the matrix visualizes the different risk levels.

IMPACT \ LIKELIHOOD	*Low*	*Medium*	*High*
Low	Weakness	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

Remark: Likelihood and Impact are categorized into three levels: H, M, and L, i.e., High, Medium and Low respectively. Severity is determined by likelihood and impact and can be classified into five categories accordingly, i.e., Critical, High, Medium, Low and Weakness

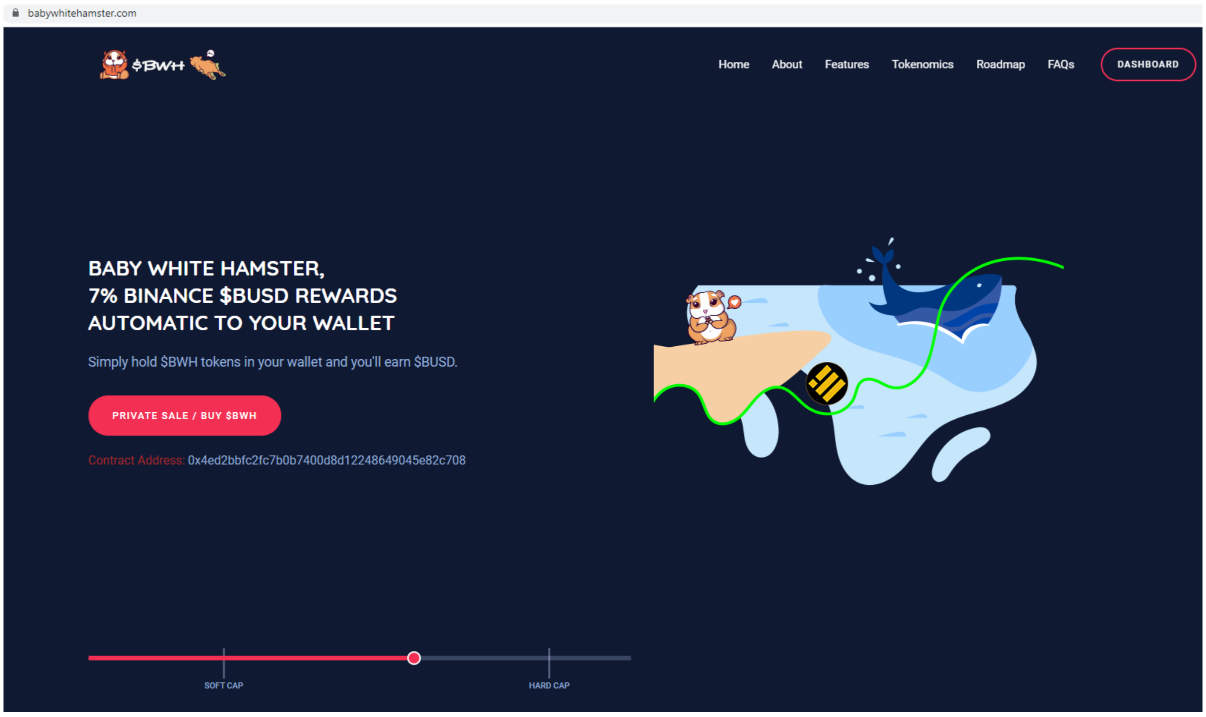
For prioritization of the vulnerabilities, we have adopted the scheme by five distinct levels for risk: Critical, High, Medium, Low, and Weakness. The risk level definitions are presented in table.

LEVEL	DESCRIPTION
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project

4 Project Overview

4.1 Communication Channels

https://babywhitehamster.com/


babywhitehamster.com

BABY WHITE HAMSTER,
7% BINANCE $BUSD REWARDS
AUTOMATIC TO YOUR WALLET

Simply hold $BWH tokens in your wallet and you'll earn $BUSD.

Contract Address: 0x4ed2bbfc2fc7b0b7400d8d12248649045e82c708

Website was registered on 23-08-2021, registration expires 23-08-2022.

Above the image is an actual snapshot of the current live website of the project.

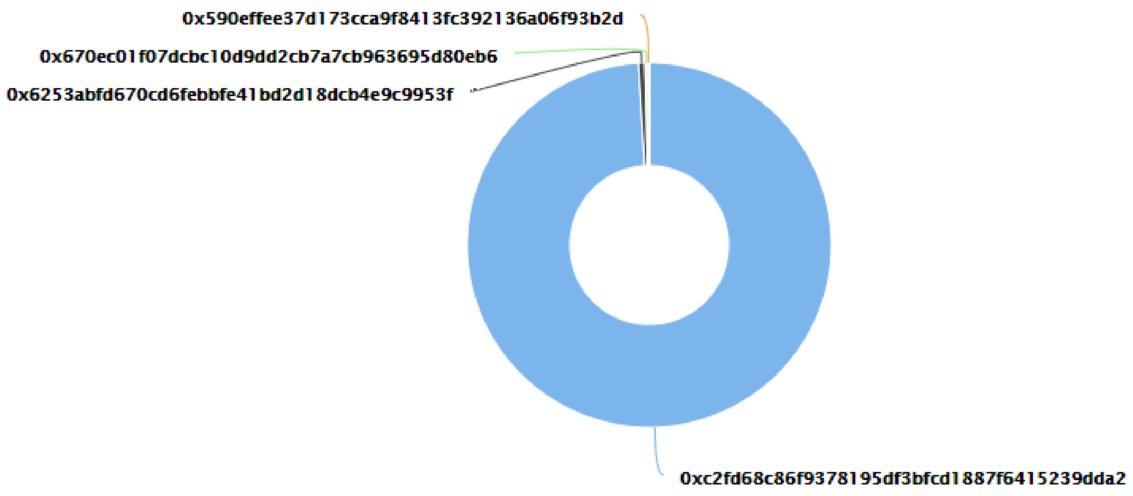
✓ Mobile Friendly	✓ 1 Social Media Networks [RISK]
✓ No JavaScript Errors	✓ < 500 Telegram Members [RISK]
✓ Visionary Roadmap	✓ No injected spam found
✓ Spell Check	✓ No popus found
✓ Valid SSL Certificate	✓ No active voice chat [RISK]
✓ White Paper	

Remark: This page contains active links

4.2 Smart Contract Details

Contract Name	Baby White Hamster
Contract Address	0x4ed2bBfc2fC7b0B7400D8D12248649045e82c708
Total Supply	100,000,000,000
Token Ticker	$BWH
Decimals	9
Token Holders	6
Transactions Count	6
Top 3 Holders Dominance	99,9%
Marketing Fee Reciever	0xc2fd68c86f9378195df3bfcd1887f6415239dda2
Uniswap V2 Pair	0xfab0686067eac5d1fe2d03cadd216f2202be5c8b
Contract Deployer Address	0xc2fd68c86f9378195df3bfcd1887f6415239dda2
Current Owner Address	0xc2fd68c86f9378195df3bfcd1887f6415239dda2

Baby White Hamster Top 10 Token Holders



Rank	Address	Quantity (Token)	Percentage
1	0xc2fd68c86f9378195df3bfcd1887f6415239dda2	99,058,030,985	99.0580%
2	0x6253abfd670cd6febbfe41bd2d18dcb4e9c9953f	557,500,000	0.5575%
3	0x670ec01f07dcbc10d9dd2cb7a7cb963695d80eb6	224,977,371	0.2250%
4	0x590effee37d173cca9f8413fc392136a06f93b2d	157,213,644	0.1572%
5	0x84826dde6a83298d30a5069d9abf1107e15c4d48	1,378,000	0.0014%
6	0xe41cb5c17273c86bea7f00c74117edccc8fb5943	900,000	0.0009%

[RISK] The contract deployer has 99% of the tokens

4.3 Contract Function Details

```
$ = payable function
# = non-constant function
[Int] = Internal
[Pub] = Public
[Prv] = Private
[Ext] = External

+ [Lib] SafeMath
   - [Int] add
   - [Int] sub
   - [Int] sub
   - [Int] mul
   - [Int] div
   - [Int] div

+ [Int] IBEP20
   - [Ext] totalSupply
   - [Ext] decimals
   - [Ext] symbol
   - [Ext] name
   - [Ext] getOwner
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ Auth
   - [Pub] #
   - [Pub] authorize #
      - modifiers: onlyOwner
   - [Pub] unauthorize #
      - modifiers: onlyOwner
   - [Pub] isOwner
   - [Pub] isAuthorized
   - [Pub] transferOwnership #
      - modifiers: onlyOwner

+ [Int] IDEXFactory
   - [Ext] createPair #

+ [Int] IDEXRouter
   - [Ext] factory
   - [Ext] WETH
   - [Ext] addLiquidity #
   - [Ext] addLiquidityETH $
   - [Ext] removeLiquidity #
   - [Ext] removeLiquidityETH #
   - [Ext] swapExactTokensForTokensSupportingFeeOnTransferTokens #
   - [Ext] swapExactETHForTokensSupportingFeeOnTransferTokens $
   - [Ext] swapExactTokensForETHSupportingFeeOnTransferTokens #

 + [Int] IDividendDistributor
   - [Ext] setDistributionCriteria #
   - [Ext] setShare #
   - [Ext] deposit $
   - [Ext] process #

 + DividendDistributor (IDividendDistributor)
   - [Pub] <Constructor> #
   - [Ext] setDistributionCriteria #
      - modifiers: onlyOwner
   - [Ext] setShare #
      - modifiers: onlyOwner
   - [Ext] deposit $
      - modifiers: onlyOwner
   - [Ext] process #
      - modifiers: onlyOwner
   - [Int] shouldDistribute
   - [Int] distributeDistribute
   - [Ext] claimDividend #
   - [Pub] getUnpaidEarnings
   - [Int] getCumulativeDividends
   - [Int] addShareholder #
   - [Int] removeShareholder #

 + BabyWhiteHamster (BEP20, Auth)
   - [Pub] <Constructor> #
      - modifiers: Auth
   - [Ext] <Fallback> $
   - [Ext] totalSupply
   - [Ext] decimals
   - [Ext] symbol
   - [Ext] name
   - [Ext] getOwner
   - [Pub] balanceOf
   - [Pub] allowance
   - [Pub] approve #
   - [Ext] approveMax #
   - [Ext] transfer #
   - [Int] transferFrom #
   - [Int] transferFrom #
   - [Int] _basicTransfer #
   - [Int] checkTxLimit
   - [Int] shouldTakeFee
   - [Pub] getTotalFee
```

```
- [Pub] getMultipliedFee
- [Int] takeFee #
- [Int] shouldSwapBack #
- [Int] swapBack #
  - modifiers: swapping
- [Int] shouldAutoBuyback
  - modifiers: authorized
- [Ext] triggerBuybackMultiplier #
  - modifiers: authorized
- [Ext] distributorDeposit $
  - modifiers: authorized
- [Ext] clearStuckBalance $
  - modifiers: onlyOwner
- [Ext] clearBuybackMultiplier
  - modifiers: authorized
- [Int] triggerAutoBuyback #
- [Int] buyTokens #
  - modifiers: swapping
- [Ext] setAutoBuybackSettings #
  - modifiers: authorized
- [Ext] setBuybackMultiplierSettings #
  - modifiers: authorized
- [Int] launched
- [Int] launch #
- [Ext] setTxLimit #
  - modifiers: authorized
- [Ext] setIsDividendExempt #
  - modifiers: authorized
- [Ext] setIsFeeExempt #
  - modifiers: authorized
- [Ext] setIsBlacklisted #
  - modifiers: authorized
- [Ext] setIsTxLimitExempt #
  - modifiers: authorized
- [Ext] setFees #
  - modifiers: authorized
- [Ext] setFeeReceivers #
  - modifiers: authorized
- [Ext] setSwapBackSettings #
  - modifiers: authorized
- [Ext] setTargetLiquidity #
  - modifiers: authorized
- [Ext] setDistributionCriteria #
  - modifiers: authorized
- [Ext] setDistributorSettings #
  - modifiers: authorized
- [Pub] getCirculatingSupply
- [Pub] getLiquidityBacking
- [Pub] isOverLiquified #
```

4.4 Issues Checking Status

CHECKING ITEM	NOTES	RESULT
Arbitrary Jump with Function Type Variable	N / A	PASS
Arithmetic Accuracy Deviation	N / A	PASS
Assert Violation	N / A	PASS
Authorization through tx.origin	N / A	PASS
Business Logic	N / A	PASS
Code with No Effects	N / A	PASS
Critical Solidity Compiler	N / A	PASS
Delegatecall to Untrusted Callee	N / A	PASS
Design Logic	N / A	PASS
DoS with Block Gas Limit	N / A	PASS
DoS with Failed Call	N / A	PASS
Function Default Visibility	N / A	PASS
Hash Collisions With MVLA	N / A	PASS
Incorrect Constructor Name	N / A	PASS
Incorrect Inheritance Order	N / A	PASS
Integer Overflows and Underflows	N / A	PASS
Lack of Proper Signature Verification	N / A	PASS
Message Call with Hardcoded Gas Amount	N / A	PASS
Missing Protection Against SRA	N / A	PASS
Presence of Unused Variables	N / A	PASS
Reentrancy	N / A	PASS

CHECKING ITEM	NOTES	RESULT
Requirement Violation	N / A	PASS
Right-To-Left-Override Control Character	N / A	PASS
Shadowing State Variables	N / A	PASS
Signature Malleability	N / A	PASS
State Variable Default Visibility	N / A	PASS
Timestamp Dependence	N / A	PASS
Transaction Order Dependence	N / A	PASS
Typographical Error	N / A	PASS
Unencrypted Private Data On-Chain	N / A	PASS
Unexpected Ether balance	N / A	PASS
Uninitialized Storage Pointer	N / A	PASS
Use of Deprecated Solidity Functions	N / A	PASS
Weak Sources of Randomness From CA	N / A	PASS
Write to Arbitrary Storage Location	N / A	PASS

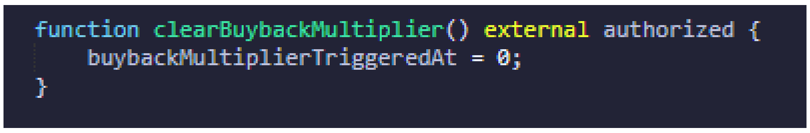
Remark: To evaluate the risk, we go through a list of check items and each would be labeled with a severity category. For one check item, if our tool or analysis does not identify any issue, the contract is considered safe regarding the check item

4.5 Detailed Findings Information

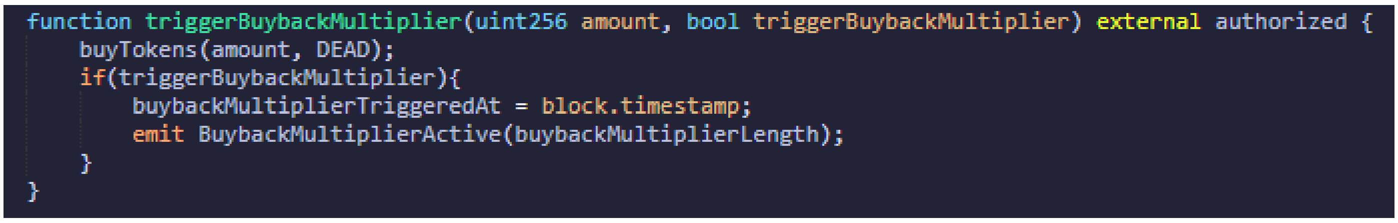
[RISK] Owner Privileges (in the period when the owner is not renounced)

The contract contains the following privileged functions that are restricted by the onlyOwner and 'authorized'.

- The owner of the contract can clean buyback multiplier and call triggerBuybackMultiplier that's initiate buyback.

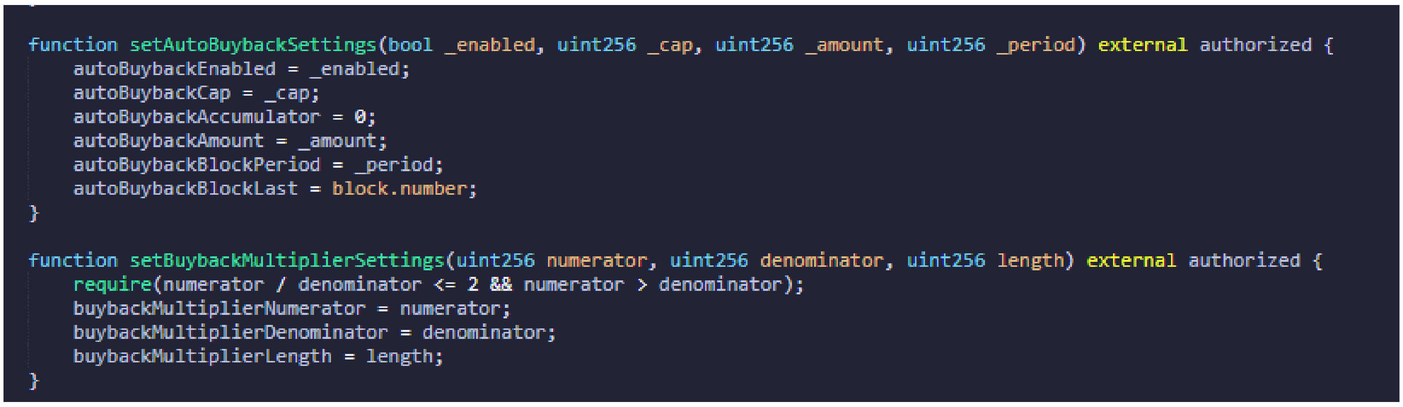


```
function clearBuybackMultiplier() external authorized {
    buybackMultiplierTriggeredAt = 0;
}
```



```
function triggerBuybackMultiplier(uint256 amount, bool triggerBuybackMultiplier) external authorized {
    buyTokens(amount, DEAD);
    if(triggerBuybackMultiplier){
        buybackMultiplierTriggeredAt = block.timestamp;
        emit BuybackMultiplierActive(buybackMultiplierLength);
    }
}
```

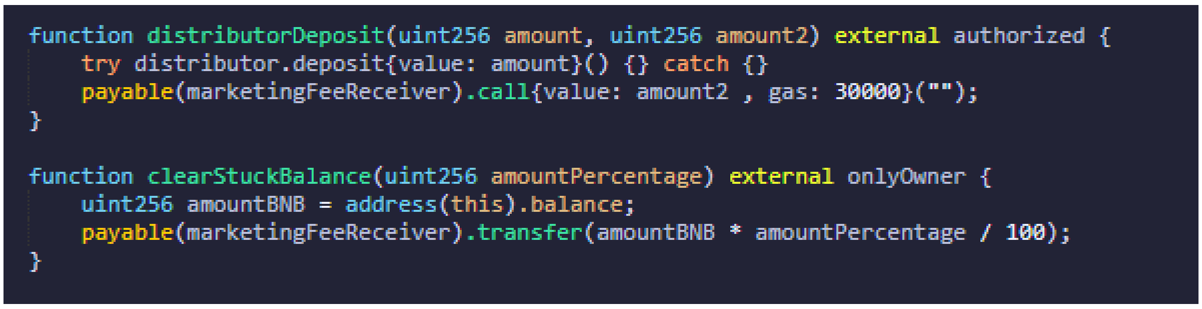
- The owner can change auto buyback settings and buyback multiplier settings.



```
function setAutoBuybackSettings(bool _enabled, uint256 _cap, uint256 _amount, uint256 _period) external authorized {
    autoBuybackEnabled = _enabled;
    autoBuybackCap = _cap;
    autoBuybackAccumulator = 0;
    autoBuybackAmount = _amount;
    autoBuybackBlockPeriod = _period;
    autoBuybackBlockLast = block.number;
}

function setBuybackMultiplierSettings(uint256 numerator, uint256 denominator, uint256 length) external authorized {
    require(numerator / denominator <= 2 && numerator > denominator);
    buybackMultiplierNumerator = numerator;
    buybackMultiplierDenominator = denominator;
    buybackMultiplierLength = length;
}
```

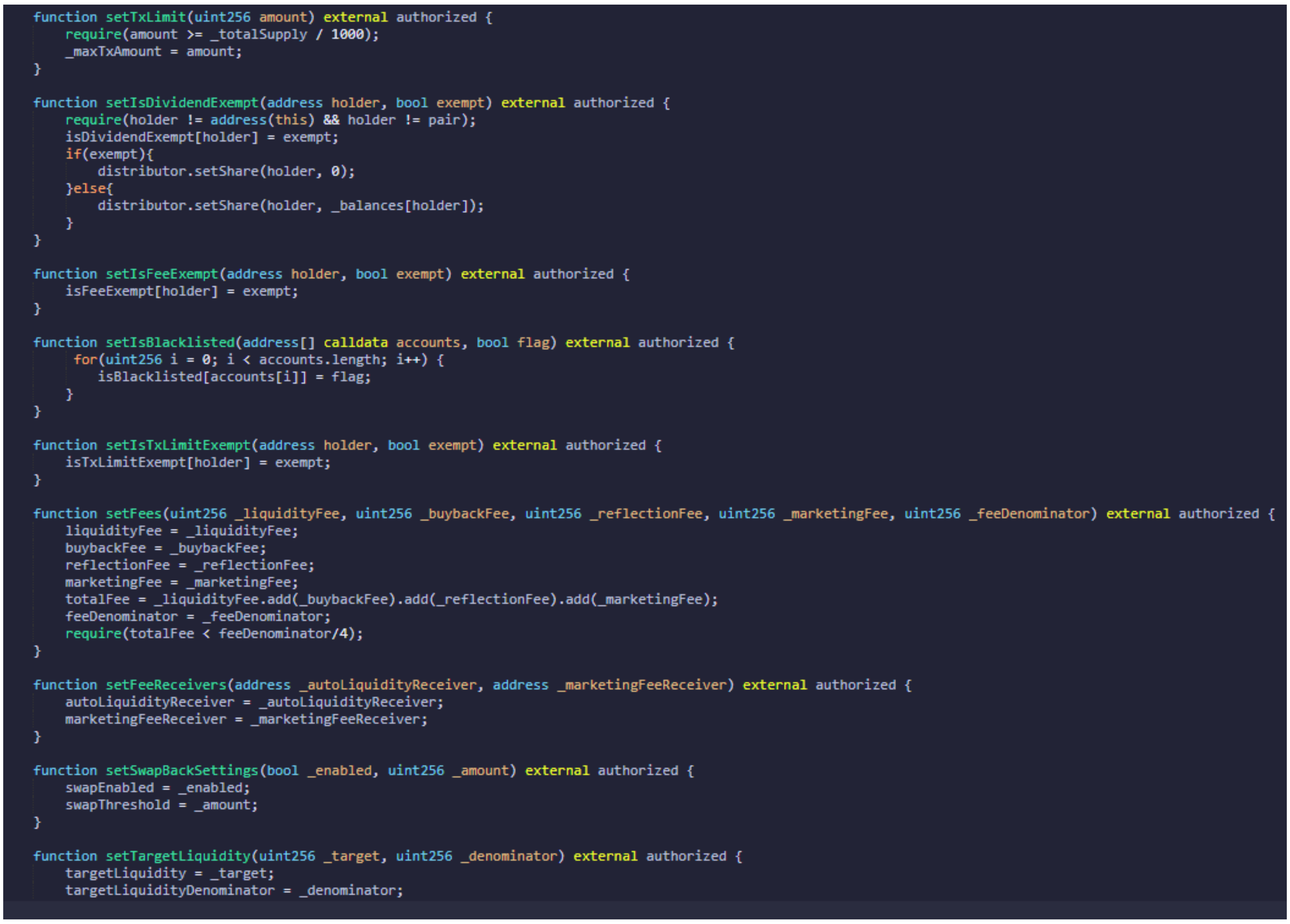
- The owner can change distribution criteria.



```
function distributorDeposit(uint256 amount, uint256 amount2) external authorized {
    try distributor.deposit{value: amount}() {} catch {}
    payable(marketingFeeReceiver).call{value: amount2 , gas: 30000}("");
}

function clearStuckBalance(uint256 amountPercentage) external onlyOwner {
    uint256 amountBNB = address(this).balance;
    payable(marketingFeeReceiver).transfer(amountBNB * amountPercentage / 100);
}
```

- The owner can change max wallet size, sell multiplier, target liquidity values, fees and fee receivers.



```
function setTxLimit(uint256 amount) external authorized {
    require(amount >= _totalSupply / 1000);
    _maxTxAmount = amount;
}

function setIsDividendExempt(address holder, bool exempt) external authorized {
    require(holder != address(this) && holder != pair);
    isDividendExempt[holder] = exempt;
    if(exempt){
        distributor.setShare(holder, 0);
    }else{
        distributor.setShare(holder, _balances[holder]);
    }
}

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

function setIsBlacklisted(address[] calldata accounts, bool flag) external authorized {
     for(uint256 i = 0; i < accounts.length; i++) {
        isBlacklisted[accounts[i]] = flag;
    }
}

function setIsTxLimitExempt(address holder, bool exempt) external authorized {
    isTxLimitExempt[holder] = exempt;
}

function setFees(uint256 _liquidityFee, uint256 _buybackFee, uint256 _reflectionFee, uint256 _marketingFee, uint256 _feeDenominator) external authorized {
    liquidityFee = _liquidityFee;
    buybackFee = _buybackFee;
    reflectionFee = _reflectionFee;
    marketingFee = _marketingFee;
    totalFee = _liquidityFee.add(_buybackFee).add(_reflectionFee).add(_marketingFee);
    feeDenominator = _feeDenominator;
    require(totalFee < feeDenominator/4);
}

function setFeeReceivers(address _autoLiquidityReceiver, address _marketingFeeReceiver) external authorized {
    autoLiquidityReceiver = _autoLiquidityReceiver;
    marketingFeeReceiver = _marketingFeeReceiver;
}

function setSwapBackSettings(bool _enabled, uint256 _amount) external authorized {
    swapEnabled = _enabled;
    swapThreshold = _amount;
}

function setTargetLiquidity(uint256 _target, uint256 _denominator) external authorized {
    targetLiquidity = _target;
    targetLiquidityDenominator = _denominator;
```

5 Audit Result

LEVEL	ISSUES
Low	**Owner Privilegies (17)**

1. The contract utilizes SafeMath libraries along with following the ERC20 standard.

2. The owner has the ability to set and update a maximum transaction percent at any time, which will impose a limit to the number of tokens that can be transferred during any given transaction. Maximum and minimum values not set. **[RISK]**

3. There is a 'liquidity fee', a buyback fee’, a ‘reflection fee’ and a ‘marketing fee’ on all transactions for any address that participates in a transfer. The owner has the ability to modify these fees from 0 to 100 percent at any time. Maximum value not set. **[RISK]**

4. A portion of the tax fee is redistributed to existing token holders instantly and automatically at the time of each transaction.

5. The owner of the contract can exclude accounts from transfer fees and reward distribution.

6. Liquidity locking details not provided by the team.

7. Ownership has not been renounced.

5.1 Findings Summary

🟩 **Baby White Hamster**
Low Risk Level

- ✓ **No external vulnerabilities were identified within the smart contract's code**
- ✓ **We strongly recommend that the team renounces ownership**
- ✓ **Please ensure trust in the team prior to investing as they have substantial control within the ecosystem**
- ✓ **We strongly recommend that the contract owners remove errors and re-audit**

6 Disclamer

CheckPoint team issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these. For the facts that occurred or existed after the issuance, CheckPoint is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to CheckPoint by the information provider till the date of the insurance report. CheckPoint is not responsible for the background and other conditions of the project.

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), CheckPoint suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.