



AUDIT REPORT

SecureWise

SMART CONTRACT AUDIT



<https://github.com/securewise>



<https://t.me/securewise>



<https://securewise.info/>



Table of Contents

03

Disclaimer

04

Overview

05

Quick Result

06

Auditing
Approach and
Methodologies

07

Automated
Analysis

10

Inheritance
Graph

11

Contract
Summary

12

Manual Review



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 GME

Methodology: Automated Analysis, Manual Code Review

Language: Solidity

Contract Address: 0xf5f8a0D87eb8ECA15bd1BF059d9641A48d8934FF

ContractLink: <https://bscscan.com/address/0xf5f8a0D87eb8ECA15bd1BF059d9641A48d8934FF>

Network: Binance Smart Chain (BSC)

Decimals: 9

Supply: 199.724,449

Website: <https://www.babygme.com/>

Twitter: https://twitter.com/baby_gme

Telegram: <https://t.me/+dPdtoFRuh4RkZDU0>

Report Date: August 23, 2022

Quick Result

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


Owner Privileges



The owner can exclude accounts from rewards



The owner can set fees with limit up to 10%



BABY GME has succesfully **PASSED** the smart contract audit with **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

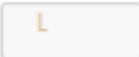

























High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

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

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

Automated Analysis

Symbol	Meaning
	Function can modify state
	Function is payable

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
L	_msgSender	Internal 		
L	_msgData	Internal 		
IBEP20	Interface			
L	totalSupply	External 		NO 
L	balanceOf	External 		NO 
L	transfer	External 		NO 
L	allowance	External 		NO 
L	approve	External 		NO 
L	transferFrom	External 		NO 
SafeMath	Library			
L	add	Internal 		
L	sub	Internal 		
L	sub	Internal 		
L	mul	Internal 		
L	div	Internal 		
L	div	Internal 		
L	mod	Internal 		
L	mod	Internal 		

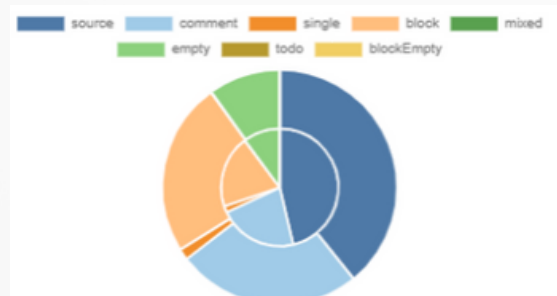
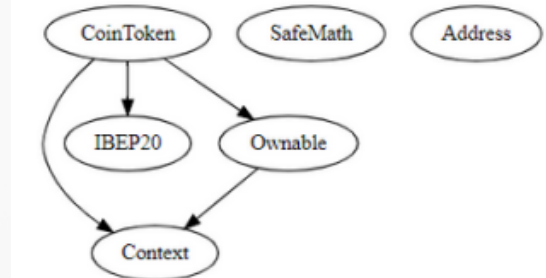
Automated Analysis

Address	Library			
L	isContract	Internal 🚫		
L	sendValue	Internal 🚫	●	
L	functionCall	Internal 🚫	●	
L	functionCall	Internal 🚫	●	
L	functionCallWithValue	Internal 🚫	●	
L	functionCallWithValue	Internal 🚫	●	
L	_functionCallWithValue	Private 🚫	●	
Ownable	Implementation	Context		
L	owner	Public 🚫		NO 🚫
L	renounceOwnership	Public 🚫	●	onlyOwner
L	transferOwnership	Public 🚫	●	onlyOwner
CoinToken	Implementation	Context, IBEP20, Ownable		
L		Public 🚫	●	NO 🚫
L	name	Public 🚫		NO 🚫
L	symbol	Public 🚫		NO 🚫
L	decimals	Public 🚫		NO 🚫
L	totalSupply	Public 🚫		NO 🚫
L	balanceOf	Public 🚫		NO 🚫
L	transfer	Public 🚫	●	NO 🚫
L	allowance	Public 🚫		NO 🚫
L	approve	Public 🚫	●	NO 🚫
L	transferFrom	Public 🚫	●	NO 🚫
L	increaseAllowance	Public 🚫	●	NO 🚫
L	decreaseAllowance	Public 🚫	●	NO 🚫
L	isExcluded	Public 🚫		NO 🚫
L	totalFees	Public 🚫		NO 🚫
L	totalBurn	Public 🚫		NO 🚫
L	totalCharity	Public 🚫		NO 🚫
L	deliver	Public 🚫	●	NO 🚫



Automated Analysis

L	reflectionFromToken	Public		NO
L	tokenFromReflection	Public		NO
L	excludeAccount	External		onlyOwner
L	includeAccount	External		onlyOwner
L	setAsCharityAccount	External		onlyOwner
L	updateFee	Public		onlyOwner
L	_approve	Private		
L	_transfer	Private		
L	_transferStandard	Private		
L	_standardTransferContent	Private		
L	_transferToExcluded	Private		
L	_excludedFromTransferContent	Private		
L	_transferFromExcluded	Private		
L	_excludedToTransferContent	Private		
L	_transferBothExcluded	Private		
L	_bothTransferContent	Private		
L	_reflectFee	Private		
L	_getValues	Private		
L	_getTBasics	Private		
L	getTTransferAmount	Private		
L	_getRBasics	Private		
L	_getRTransferAmount	Private		
L	_getRate	Private		
L	_getCurrentSupply	Private		
L	_sendToCharity	Private		
L	removeAllFee	Private		
L	restoreAllFee	Private		
L	_getTaxFee	Private		

Inheritance Graph



Contract Summary

Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
5	1	828	755	448	283	325	
5	1	828	755	448	283	325	

Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	2	1	1

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
31	0

External	Internal	Private	Pure	View
9	52	23	11	24

StateVariables

Total	 Public
24	5

Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts	
<div>^0.8.2</div>			<div>yes</div> <div>(2 asm blocks)</div>		
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRRecover	 New/Create/Create2
 TryCatch	Σ Unchecked				

Manual Review

The owner can exclude accounts from rewards

```
function excludeAccount(address account) external onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_rOwned[account] > 0) {
        _tOwned[account] = tokenFromReflection(_rOwned[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}

function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

Recommendation

Privileged roles can be include/exclude account from rewards these cause affect decentralization

The owner can set fees with limit up to 10%

```
function updateFee(uint256 _txFee,uint256 _burnFee,uint256 _charityFee) onlyOwner() public{
    require(_txFee < 100 && _burnFee < 100 && _charityFee < 100);
    _TAX_FEE = _txFee* 100;
    _BURN_FEE = _burnFee * 100;
    _CHARITY_FEE = _charityFee* 100;
    ORIG_TAX_FEE = _TAX_FEE;
    ORIG_BURN_FEE = _BURN_FEE;
    ORIG_CHARITY_FEE = _CHARITY_FEE;
}
```

Recommendation

Privileged roles can be set fees limit up 10%. Seems to reasonable limit.

Manual Review

The owner can change fee Wallet Address

```
function setAsCharityAccount(address account) external onlyOwner() {  
    FeeAddress = account;  
}
```


Recommendation

Privileged roles can be change fee wallet address. Put a **require** and check wallet address **not equal** zero wallet.

AUDIT REPORT

SecureWise

SMART CONTRACT AUDIT

 <https://github.com/securewise>
 <https://t.me/securewise>
 <https://securewise.info/>

