RUGFREECOINS

Baby Froggy Token

RugfreeCoins Verified on October 9th, 2023

Overview

✅ No mint function found, the owner cannot mint tokens after initial deployment.

✅ The owner can't set a max transaction limit

✅ The owner can't pause trading once it's enabled

❌ The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

✅ The owner can't change fees.

✅ The owner can't blacklist wallets.

✅ The owner can't set a max wallet limit

✅ The owner can't claim the contract's balance of its own token.

❗ HIGH SEVERITY ISSUES

The owner has the authority to adjust the swap threshold, allowing for changes of up to 1000% of the total supply. However, it's important to note that if the owner sets an excessively high swap threshold, it could lead to potential failures in the swap process due to a significant price impact. Such failures have the potential to halt all selling transactions.

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
    require(
        new_amount <= 1e7,
        "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
}
```

The launch taxes are initially set to a default value of 0, and the owner lacks any function to modify them subsequently. Consequently, all trades occurring within the specified dead block period will incur 0 taxes.

```
} else if (useLaunchFee) {
    feeswap = launchtax;
    feesum = launchtax;
}
```

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

Contents

Audit details

Audited project

Baby Froggy Token

Contract Address

0xe70238fe095900958186362c03e4577a1db538b4

Client contact

Baby Froggy Token Team

Blockchain

Binance Smart chain

Project website

https://babyfroggy.site

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – **please make sure to read it in full.**

ⓘ 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 **RugfreeCoins and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (RugfreeCoins) owe no duty of care towards you or any other person**, nor does RugfreeCoins 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 RugfreeCoins 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, RugfreeCoins hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against RugfreeCoins, 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.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

RugfreeCoins was commissioned by the **Baby Froggy Token Team** to perform an audit of the smart contract.

https://bscscan.com/token/0xe70238fe095900958186362c03e4577a1db538b4

This audit focuses on verifying that the smart contract is secure, resilient, and working according to the specifications.

The information in this report should be used to understand the risk exposure of the smart contract, project feasibility, and long-term sustainability, and as a guide to improving the smart contract's security posture by remediating the identified issues.

Tokenomics

▲ 2% tax when buying & selling (09/10/2023)

2% of trade goes to the Dev wallet in BNB
0% of trade goes to the Liquidity pool

Target market and the concept

- Anyone who’s interested in the Crypto space with long-term investment plans.
- Anyone who’s ready to earn a passive income by holding tokens.
- Anyone who’s interested in trading tokens.
- Anyone who’s interested in taking part in the Baby Froggy token ecosystem.
- Anyone who’s interested in taking part in the future plans of Baby Froggy Token.
- Anyone who’s interested in making financial transactions with any other party using Baby Froggy Token as the currency.

Potential to grow with score points

⚡ Project efficiency	8 / 10
🌟 Project uniqueness	7 / 10
📊 Information quality	8 / 10
👌 Service quality	8 / 10
💻 System quality	8 / 10
🌍 Impact on the community	8 / 10
💼 Impact on the business	9 / 10
🔮 Preparing for the future	8 / 10
🔒 Smart contract security	7 / 10
🛠️ Smart contract functionality assessment	6 / 10
🏆 **Total Score**	**7.7** / 10

Contract details

Token contract details for 9th of October 2023

Contract name	**BabyFroggy**
Contract address	**0xe70238fE095900958186362c03e4577A1Db538B4**
Token supply	**1,000,000**
Token ticker	**BabyFroggy**
Decimals	**18**
Token holders	**1**
Transaction count	**1**
Contract deployer address	**0xC084bAe0ea6dd20c90D4E42eebfA90e55CfD0Ec7**
Contract’s current owner address	**0xC084bAe0ea6dd20c90D4E42eebfA90e55CfD0Ec7**

Contract code function details

№	Category	Item	Result
1	Coding conventions	ERC20 Token standards	PASS
		Compile errors	PASS
		Compiler version security	PASS
		Visibility specifiers	PASS
		Gas consumption	LOW
		SafeMath features	PASS
		Fallback usage	PASS
		tx.origin usage	PASS
		Deprecated items	PASS
		Redundant code	PASS
		Overriding variables	PASS
2	Function call audit	Authorization of function call	PASS
		Low level function (call/delegate call) security	PASS
		Returned value security	PASS
		Self destruct function security	PASS
3	Business security & centralisation	Access control of owners	HIGH
		Business logics	HIGH
		Business implementation	PASS
4	Integer overflow/underflow		PASS
5	Reentrancy		PASS
6	Exceptional reachable state		PASS
7	Transaction ordering dependence		PASS
8	Block properties dependence		PASS
9	Pseudo random number generator (PRNG)		PASS
10	DoS (Denial of Service)		PASS
11	Token vesting implementation		PASS
12	Fake deposit		PASS
13	Event security		PASS

Contract description table

The below table represents the summary of the contracts and methods in the token contract. We scanned the whole contract and listed down all the Interfaces, functions, and implementations with their visibility and mutability.

Contract	**Type**	**Bases**		
L	Function Name	Visibility	Mutability	Modifiers
Context	**Implementation**			
L	_msgSender	Internal 🔒		
L	_msgData	Internal 🔒		
IERC20	**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 ❗
IERC20 Metadata	**Interface**	**IERC20**		
L	name	External ❗		NO ❗
L	symbol	External ❗		NO ❗
L	decimals	External ❗		NO ❗
ERC20	**Implementation**	**Context, IERC20, IERC20 Metadata**		
L		Public ❗	🛑	NO ❗

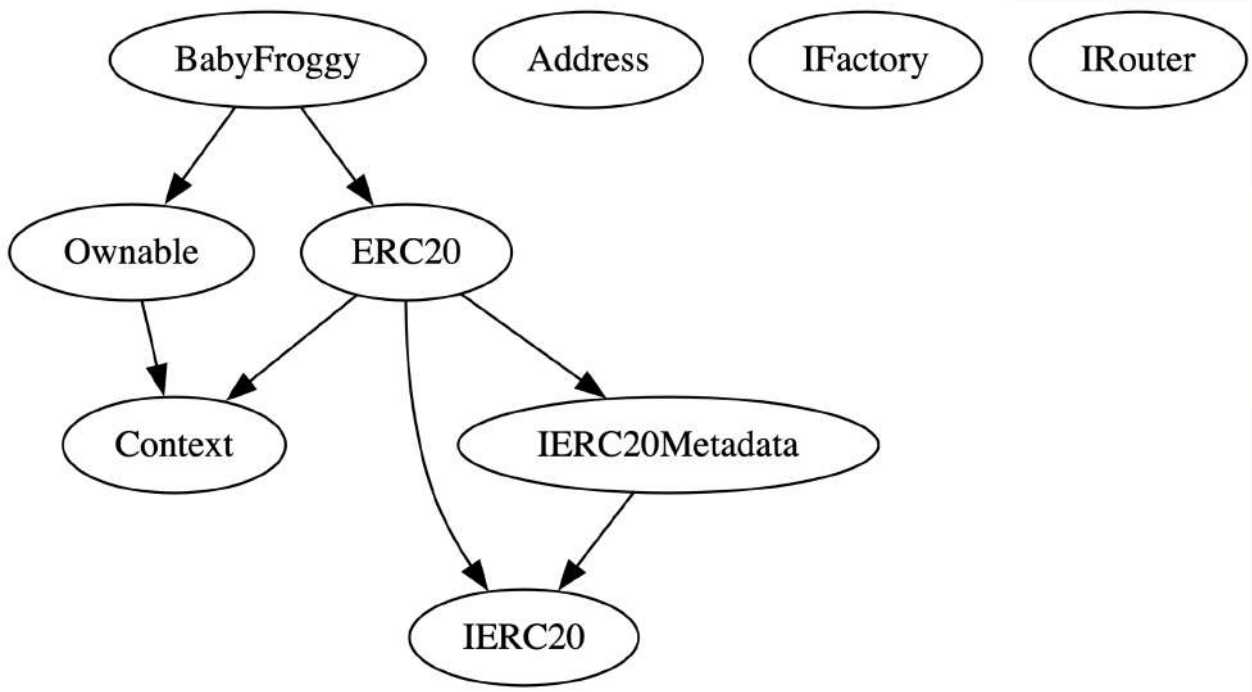
L	name	Public ❗		NO ❗
L	symbol	Public ❗		NO ❗
L	decimals	Public ❗		NO ❗
L	totalSupply	Public ❗		NO ❗
L	balanceOf	Public ❗		NO ❗
L	transfer	Public ❗	🛑	NO ❗
L	allowance	Public ❗		NO ❗
L	approve	Public ❗	🛑	NO ❗
L	transferFrom	Public ❗	🛑	NO ❗
L	increaseAllowance	Public ❗	🛑	NO ❗
L	decreaseAllowance	Public ❗	🛑	NO ❗
L	_transfer	Internal 🔒	🛑	
L	_tokengeneration	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
Address	**Library**			
L	sendValue	Internal 🔒	🛑	
Ownable	**Implementation**	**Context**		
L		Public ❗	🛑	NO ❗
L	owner	Public ❗		NO ❗
L	renounceOwnership	Public ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
L	_setOwner	Private 🔐	🛑	
IFactory	**Interface**			
L	createPair	External ❗	🛑	NO ❗
IRouter	**Interface**			

└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
BabyFroggy	**Implementation**	**ERC20, Ownable**		
└		Public ❗	🛑	ERC20
└	approve	Public ❗	🛑	NO ❗
└	transferFrom	Public ❗	🛑	NO ❗
└	increaseAllowance	Public ❗	🛑	NO ❗
└	decreaseAllowance	Public ❗	🛑	NO ❗
└	transfer	Public ❗	🛑	NO ❗
└	_transfer	Internal 🔒	🛑	
└	Liquify	Private 🔐	🛑	lockTheSwap
└	swapTokensForETH	Private 🔐	🛑	
└	addLiquidity	Private 🔐	🛑	
└	updateLiquidityProvide	External ❗	🛑	onlyOwner
└	updateLiquidityTreshhold	External ❗	🛑	onlyOwner
└	EnableTrading	External ❗	🛑	onlyOwner
└	updatedeadline	External ❗	🛑	onlyOwner
└	updateDevWallet	External ❗	🛑	onlyOwner
└	updateTax	External ❗	🛑	onlyOwner
└	updateExemptFee	External ❗	🛑	onlyOwner
└	bulkExemptFee	External ❗	🛑	onlyOwner
└	rescueBNB	External ❗	🛑	onlyOwner
└	rescueBEP20	External ❗	🛑	onlyOwner
└		External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Inheritance Hierarchy



Security issue checking status

❖ High severity issues

The owner has the authority to adjust the swap threshold, allowing for changes of up to 1000% of the total supply. However, it's important to note that if the owner sets an excessively high swap threshold, it could lead to potential failures in the swap process due to a significant price impact. Such failures have the potential to halt all selling transactions.

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
    require(
        new_amount <= 1e7,
        "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
}
```

The launch taxes are initially set to a default value of 0, and the owner lacks any function to modify them subsequently. Consequently, all trades occurring within the specified dead block period will incur 0 taxes.

```
} else if (useLaunchFee) {
    feeswap = launchtax;
    feesum = launchtax;
}
```

The owner must enable trade for the holders, if trading remains disabled, no one would be able to buy and sell.

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

❖ Medium severity issues

No medium severity issues found

❖ Low severity issues

The variables feeSwap and feesum consistently hold the same value. Therefore, there is no necessity to maintain two separate variables for handling this situation. The presence of extra variables consumes additional gas unnecessarily.

```
else if (recipient == pair && !useLaunchFee) {
    feeswap = sellTaxes.liquidity + sellTaxes.dev;
    feesum = feeswap;
    currentTaxes = sellTaxes;
} else if (!useLaunchFee) {
    feeswap = taxes.liquidity + taxes.dev;
    feesum = feeswap;
    currentTaxes = taxes;
} else if (useLaunchFee) {
    feeswap = launchtax;
    feesum = launchtax;
}
```

Since feeSwap and feesum have the same value for "fee," and feeAmount also holds the same value, there is no need to calculate feeAmount separately; it can be used directly for this purpose.

```
        //rest to recipient
        super._transfer(sender, recipient, amount - fee);
        if (fee > 0) {
            //send the fee to the contract
            if (feeswap > 0) {
                uint256 feeAmount = (amount * feeswap) / 100;
                super._transfer(sender, address(this), feeAmount);
            }
        }
```

When excluding multiple wallets from fees, the owner has the flexibility to input any number of wallets simultaneously. However, it's important to note that if the owner enters a large array of wallets, the transaction may fail due to the gas limit.

```
    function bulkExemptFee(
        address[] memory accounts,
        bool state
    ) external onlyOwner {
        for (uint256 i = 0; i < accounts.length; i++) {
            exemptFee[accounts[i]] = state;
        }
    }
```

Owner privileges

- Owner can enable/disable swapping

```
function updateLiquidityProvide(bool state) external onlyOwner {
    providingLiquidity = state;
}
```

- Owner can change the swap threshold up-to 1000%

```
function updateLiquidityTreshhold(uint256 new_amount) external onlyOwner {
    require(
        new_amount <= 1e7,
        "Swap threshold amount should be lower or equal to 1% of tokens"
    );
    tokenLiquidityThreshold = new_amount * 10 ** decimals();
}
```

- Owner can enable trading, once enabled can not disable it again

```
function EnableTrading() external onlyOwner {
    require(!tradingEnabled, "Cannot re-enable trading");
    tradingEnabled = true;
    providingLiquidity = true;
    genesis_block = block.number;
}
```

❖ Owner can change the number of dead blocks up-to 4 before enabling trades (all trades within dead blocks will take launch taxes)

```
function updatedeadline(uint256 _deadline) external onlyOwner {
    require(!tradingEnabled, "Can't change when trading has started");
    require(_deadline < 5, "Deadline should be less than 5 Blocks");
    deadline = _deadline;
}
```

❖ Owner can change all buy and sell fees total fees up-to 15% (buy tax up-to 5% and sell tax up-to 10%)

```
function updateTax(
    uint256 buyDevTax,
    uint256 buyLiquidityTax,
    uint256 sellDevTax,
    uint256 sellLiquidityTax
) external onlyOwner {
    require(
        (buyDevTax + buyLiquidityTax) <= 5,
        "Can't set tax greater than 5%"
    );
    require(
        (sellDevTax + sellLiquidityTax) <= 10,
        "Can't set tax greater than 10%"
    );
    taxes = Taxes(buyDevTax, buyLiquidityTax);
    sellTaxes = Taxes(sellDevTax, sellLiquidityTax);
}
```

❖ Owner can include/exclude wallets from fees

```
function updateExemptFee(address _address, bool state) external onlyOwner {
    exemptFee[_address] = state;
}
```

❖ Owner can include/exclude multiple wallets from fees

```
function bulkExemptFee(
    address[] memory accounts,
    bool state
) external onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        exemptFee[accounts[i]] = state;
    }
}
```

❖ Owner can get contract BNB balance to owner wallet

```
function rescueBNB(uint256 weiAmount) external onlyOwner {
    payable(owner()).transfer(weiAmount);
}
```

❖ Owner can get any BEP20 tokens from the contract (can not get native tokens)

```
function rescueBEP20(address tokenAdd, uint256 amount) external onlyOwner {
    require(
        tokenAdd != address(this),
        "Owner can't claim contract's balance of its own tokens"
    );
    IERC20(tokenAdd).transfer(owner(), amount);
}
```

❖ Owner can change dev wallet address

```
function updateDevWallet(address newWallet) external onlyOwner {
    require(newWallet != address(0), "Fee Address cannot be zero address");
    devWallet = newWallet;
}
```

Audit conclusion

RugFreeCoins team has performed in-depth testing, line-by-line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status:	PASS
Smart contract security Status:	HIGH ISSUES
Number of risk issues:	6
Solidity code functional issue level:	PASS
Number of owner privileges:	10
Centralization risk correlated to the active owner:	HIGH
Smart contract active ownership:	ACTIVE