expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

BabyDummyToken Presale

Contract Address

0xFBe8F9314ca46aDBA6Ad137985d62B4fDdDA397A

Audit Summary

Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result: **PASSED**

Ownership: NOT **RENOUNCED**

KYC Verification: Not done till date of audit

Audit Date: 02/07/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.

Contract Review

Contract Name	**BABYTOKEN**
Compiler Version	**v0.8.4+commit.c7e474f2**
Optimization	**Yes with 200 runs**
License	**MIT license**
Explorer	**https://bscscan.com/address/0xFBe8F9314ca46aDBA6Ad137985d62B4fDdDA397A#code**
Symbol	**BabyD**
Decimals	**18**
Total Supply	**1,000,000,000,000,000**
Domain	**https://www.babydummytoken.com/**

Project Review

Token Name: FitApe

Web Site: https://www.babydummytoken.com/

Twitter: https://twitter.com/babydummy_token

Telegram: https://t.me/babyd_token

Contract Address:
0xFBe8F9314ca46aDBA6Ad137985d62B4fDdDA397A

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

Low-Risk

3 low-risk code issues found

Medium-Risk

0 medium-risk code issues found

High-Risk

0 high-risk code issues found

Low-Risk

1)Contract contains Reentrancy vulnuerabilities

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal override {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        if (amount == 0) {
            super._transfer(from, to, 0);
            return;
        }

        uint256 contractTokenBalance = balanceOf(address(this));

        bool canSwap = contractTokenBalance >= swapTokensAtAmount;

        if (
            canSwap &&
            !swapping &&
            !automatedMarketMakerPairs[from] &&
            from != owner() &&
            to != owner()
        ) {
            swapping = true;

            uint256 marketingTokens = contractTokenBalance
                .mul(marketingFee)
                .div(totalFees);
            swapAndSendToFee(marketingTokens);

            uint256 swapTokens = contractTokenBalance.mul(liquidityFee).div(
                totalFees
            );
            swapAndLiquify(swapTokens);

            uint256 sellTokens = balanceOf(address(this));
            swapAndSendDividends(sellTokens);

            swapping = false;
        }
```

Recommendation

Apply the check-effects-interaction pattern

2) No zero address validation for some functions

Detect missing zero address validation.

```
function setMarketingWallet(address payable wallet) external onlyOwner {
        _marketingWalletAddress = wallet;
    }
```

Recommendation

Check that the new address is not zero.

3) **Function that send Ether to arbitary destination.**
Unprotected call to a function sending Ether to arbitary address.

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            address(0),
            block.timestamp
        );
    }
```

Recommendation

Ensure that an arbitary user cannot withdraw unauthorized funds

Audit Summary

```
Compiled with solc
Number of lines: 3153 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 26 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 43
Number of informational issues: 63
Number of low issues: 3
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC20, ERC2612
```

Name	# functions	ERCS	ERC20 info	Complex code	Features
SafeMath	13			No	
Clones	4			No	Assembly
IUniswapV2Factory	8			No	
IUniswapV2Router02	24			No	Receive ETH
IUniswapV2Pair	27	ERC20,ERC2612	∞ Minting Approve Race Cond.	No	
SafeMathInt	7			No	
SafeMathUint	1			No	
IterableMapping	6			No	
BABYTOKENDividendTracker	71	ERC20	No Minting Approve Race Cond.	Yes	Tokens interaction Upgradeable
BABYTOKEN	72	ERC20	No Minting Approve Race Cond.	Yes	Receive ETH Send ETH Tokens interaction

Manual Audit (Contract Function)

```
contract BABYTOKENDividendTracker is OwnableUpgradeable, DividendPayingToken {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using IterableMapping for IterableMapping.Map;

    IterableMapping.Map private tokenHoldersMap;
    uint256 public lastProcessedIndex;

    mapping(address => bool) public excludedFromDividends;

    mapping(address => uint256) public lastClaimTimes;

    uint256 public claimWait;
    uint256 public minimumTokenBalanceForDividends;

    event ExcludeFromDividends(address indexed account);
    event ClaimWaitUpdated(uint256 indexed newValue, uint256 indexed oldValue);

    event Claim(
        address indexed account,
        uint256 amount,
        bool indexed automatic
    );

    function initialize(
        address rewardToken_,
        uint256 minimumTokenBalanceForDividends_
    ) external initializer {
        DividendPayingToken.__DividendPayingToken_init(
            rewardToken_,
            "DIVIDEND_TRACKER",
            "DIVIDEND_TRACKER"
        );
        claimWait = 3600;
        minimumTokenBalanceForDividends = minimumTokenBalanceForDividends_;
    }

    function _transfer(
        address,
        address,
        uint256
    ) internal pure override {
        require(false, "Dividend_Tracker: No transfers allowed");
    }
```

Important Points To Consider

✓ The owner cannot stop Trading.

✓ Verified contract source

✗ Token is sellable (not a honeypot) at this time

✗ Ownership renounced or source does not contain an owner contract

✗ Source does not contain a fee modifier

✓ Owner/creator wallet contains less than 10% of circulating token supply (4.98%)

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co