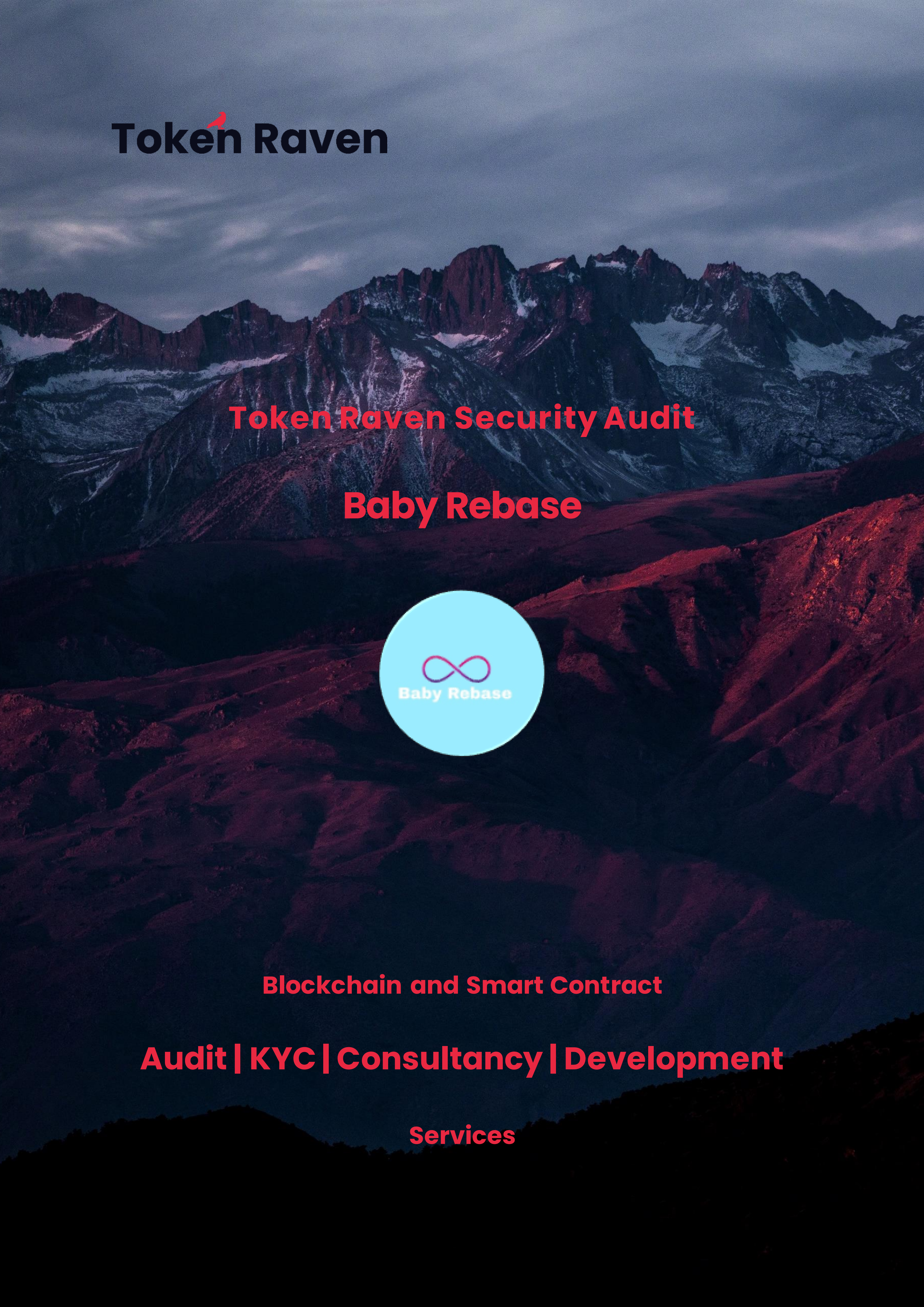


Token Raven

Token Raven Security Audit

Baby Rebase

Baby Rebase

Blockchain and Smart Contract

Audit | KYC | Consultancy | Development

Services

Summary

Token Raven has performed a combination of manual and software analysis of the smart contracts provided by the project team. Each of them has been analyzed for the most common and exploits, vulnerabilities, and manipulation attacks.

For Baby Rebase we have come to the following conclusion.

- The solidity source code has **Low Severity**
- The smart contract has an **Active Ownership**
- The centralization risk is therefore **high**
- The only Owner privileges are **changing fees, exclude wallets from fee and disable trading etc.**

All the content provided in the Token Raven audit reports are for general information and should not be used as financial advice nor a reason to invest in a project. Whilst the rigorousness of our audit should improve the safety of investing in a project over an unaudited counterpart, the Token Raven team is not responsible in the event of a fraud or scam committed by any of our partners. We implore everyone to always do your own due diligence.

The authenticity of this document can be verified on the Token Raven GitHub https://github.com/tokenraven

This audit was completed on: **11-05-2022**

Table of Contents

Audit Summary 4
Provided Description 4
Audit Process 5
Common Risks 6
Software Analysis 7
Static 7
Function Signatures 10
SWC Attacks 12
Inheritance graph 15
Call Graph 15
Manual Analysis 16
Owner Privileges 16
Use of block.timestamp 17
Risk Summary 18
Conclusion 19
Disclaimer 19
About 21

Audit Summary

Provided Description

Baby Rebase is the first ever Rebase contract combined with 5% ADA rewards. The rebase contract will be used to push the price through the roof to an pegged price of 1.337.000$ The sky is the limit.

Website: https://geffenna.wixsite.com/babyrebase

Project Name	Baby Rebase
Platform	BNB Chain
Language	Solidity
Contract Address	0xc1df7fdB9923D3EA30eD6c9BDAfb46a5618191E5
Compiler Version	0.7.4
Token Ticker	BR
Total Supply	1,358,536,925,812,500
Decimals	4
Tax Fees	14%
Deployer Address	0xF8e5d009c35f9F6b292db24378dd420B3E057Ef7
Current Owner	0xF8e5d009c35f9F6b292db24378dd420B3E057Ef7

Link to contract:

https://bscscan.com/address/0xc1df7fdb9923d3ea30ed6c9bdafb46a5618191e5#code

Audit Process

Our rigorous audit process is as follows.

Step 1

The smart contract is tested on the Testnet of the corresponding network as well as using software analysis tools to make sure that everything is functioning. If we notice any issues, we will provide recommendations to get these issues resolved.

Step 2

Here we check whether there are functions in the contract that are only for the owner of the contract, which therefore entail privileges. If we find this type of feature, it will be communicated to our partners as we recommend this is to be changed.

Step 3

If our partner does not want to make changes to these owner-only functions, we recommend that you verify through a KYC verification with us. We then indicate during the audit that these functions are included in the contract, but that the owner has been verified at Token Raven.

Step 4

Once we check the contract line by line in the auditing process and our recommendations for problem fixes are implemented, this will be communicated to the project team.

Step 5

If everything is in order after the previous 4 steps, the audit report including the certificate will be displayed on our website and GitHub. In the report, all our comments are divided into categories, ranging from low risk to high risk labels.

Common Risks

The cold hard logic of smart contracts allows them to facilitate the most secure and trustworthy kinds of transactions to date. This infallible ability to always execute the provided code does mean however, that the code should be free of errors in order to prevent security hazards.

The top ten Smart Contract Risks (SCR) fall into three categories

Operational Risks

SCR-1: Super User Account or Privilege Management

SCR-2: Blacklisting and Burning Functions

SCR-3: Contract Logic or Asset Configuration can be arbitrarily changed

SCR-4: Self-Destruct Functions

SCR-5: Minting Functions

Implementation Risks

SCR-6: Rolling Your Own Crypto and Unique Contract Logic

SCR-7: Unauthorized Transfers

SCR-8: Incorrect Signature Implementation or Arithmetic

Design Risks

SCR-9: Untrusted Control Flow

SCR-10: Transaction Order Dependence

All of these risks are checked for either manually or through our software analyses.

Software Analysis

Static

This static analysis creates a markdown description report comprising information about the system's files, contracts, and their functions.

Contract	Type	Bases		
L	**Function name**	**Visibility**	**Mutability**	**Modifiers**

Legend:

Symbol	Definition
🛑	Function can be modified
💵	Function is payable
🔒	Function is locked
🔐	Function can be accessed
❗	Important functionality

```
| **SafeMath** | Library |  |||
| └ | add | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
||||||
| **SafeMathInt** | Library |  |||
| └ | mul | Internal 🔒 |   | |
| └ | div | Internal 🔒 |   | |
| └ | sub | Internal 🔒 |   | |
| └ | add | Internal 🔒 |   | |
| └ | abs | Internal 🔒 |   | |
||||||
| **IBEP20** | Interface |  |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **Auth** | Implementation |  |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | authorize | Public ❗ | 🛑  | onlyOwner |
| └ | unauthorize | Public ❗ | 🛑  | onlyOwner |
| └ | isOwner | Public ❗ |   |NO❗ |
| └ | isAuthorized | Public ❗ |   |NO❗ |
| └ | transferOwnership | Public ❗ | 🛑  | onlyOwner |
||||||
| **IDEXFactory** | Interface |  |||
| └ | createPair | External ❗ | 🛑  |NO❗ |
||||||
| **InterfaceLP** | Interface |  |||
| └ | sync | 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 |||
| └ | <Constructor> | Public ❗ | 🛑  |NO❗ |
| └ | setDistributionCriteria | External ❗ | 🛑  | onlyToken |
| └ | setShare | External ❗ | 🛑  | onlyToken |
| └ | deposit | External ❗ |  💵 | onlyToken |
| └ | process | External ❗ | 🛑  | onlyToken |
| └ | shouldDistribute | Internal 🔒 |   | |
| └ | distributeDividend | Internal 🔒 | 🛑  | |
| └ | claimDividend | External ❗ | 🛑  |NO❗ |
| └ | getUnpaidEarnings | Public ❗ |   |NO❗ |
| └ | getCumulativeDividends | Internal 🔒 |   | |
| └ | addShareholder | Internal 🔒 | 🛑  | |
| └ | removeShareholder | Internal 🔒 | 🛑  | |
||||||
| **BabyRebase** | Implementation | IBEP20, Auth |||
| └ | rebase_percentage | Public ❗ | 🛑  | onlyOwner |
| └ | rebase | Public ❗ | 🛑  | onlyMaster |
| └ | <Constructor> | Public ❗ | 🛑  | Auth |
| └ | <Receive Ether> | 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❗ |
| └ | _transferFrom | Internal 🔒 | 🛑  | |
| └ | _basicTransfer | Internal 🔒 | 🛑  | |
| └ | checkTxLimit | Internal 🔒 |   | |
| └ | shouldTakeFee | Internal 🔒 |   | |
| └ | takeFee | Internal 🔒 | 🛑  | |
| └ | shouldSwapBack | Internal 🔒 |   | |
| └ | clearStuckBalance | External ❗ | 🛑  | authorized |
| └ | clearStuckBalance_sender | External ❗ | 🛑  | authorized |
| └ | set_sell_multiplier | External ❗ | 🛑  | onlyOwner |
| └ | tradingStatus | Public ❗ | 🛑  | onlyOwner |
| └ | launchStatus | Public ❗ | 🛑  | onlyOwner |
| └ | enable_hotel_CaliforniaMode | Public ❗ | 🛑  | onlyOwner |
| └ | set_max_roomrent | Public ❗ | 🛑  | onlyOwner |
| └ | manage_houseguests | Public ❗ | 🛑  | onlyOwner |
| └ | cooldownEnabled | Public ❗ | 🛑  | onlyOwner |
| └ | swapBack | Internal 🔒 | 🛑  | swapping |
| └ | setIsDividendExempt | External ❗ | 🛑  | authorized |
| └ | setIsFeeExempt | External ❗ | 🛑  | authorized |
| └ | setIsTxLimitExempt | External ❗ | 🛑  | authorized |
| └ | setIsTimelockExempt | External ❗ | 🛑  | authorized |
| └ | setFees | External ❗ | 🛑  | authorized |
| └ | setFeeReceivers | External ❗ | 🛑  | authorized |
| └ | setSwapBackSettings | External ❗ | 🛑  | authorized |
| └ | setTargetLiquidity | External ❗ | 🛑  | authorized |
| └ | manualSync | External ❗ | 🛑  |NO❗ |
| └ | setLP | External ❗ | 🛑  | onlyOwner |
| └ | setMaster | External ❗ | 🛑  | onlyOwner |
| └ | isNotInSwap | External ❗ |   |NO❗ |
| └ | checkSwapThreshold | External ❗ |   |NO❗ |
| └ | setDistributionCriteria | External ❗ | 🛑  | authorized |
| └ | setDistributorSettings | External ❗ | 🛑  | authorized |
| └ | rescueToken | Public ❗ | 🛑  | onlyOwner |
| └ | getCirculatingSupply | Public ❗ |   |NO❗ |
| └ | getLiquidityBacking | Public ❗ |   |NO❗ |
| └ | isOverLiquified | Public ❗ |   |NO❗ |
| └ | checkMaxWalletToken | External ❗ |   |NO❗ |
| └ | checkMaxTxAmount | External ❗ |   |NO❗ |
| └ | setMaxWalletPercent_base1000 | External ❗ | 🛑  | onlyOwner |
| └ | setMaxTxPercent_base1000 | External ❗ | 🛑  | onlyOwner |
| └ | multiTransfer | External ❗ | 🛑  | onlyOwner |
| └ | multiTransfer_fixed | External ❗ | 🛑  | onlyOwner |
```

Function Signatures

Solidity creates function signatures for the EVM by hashing the function name with keccak256(), which is a hashing function. This function returns a hex which is then converted into bytes4. It thus converts a function from hex to an array of 4, 8 byte numbers.

```
Sighash   |   Function Signature
========================
43509138  =>  div(int256,int256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
bbe93d91  =>  mul(int256,int256)
adefc37b  =>  sub(int256,int256)
a5f3c23b  =>  add(int256,int256)
1b5ac4b5  =>  abs(int256)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
```

```
b6a5d7de  =>  authorize(address)
f0b37c04  =>  unauthorize(address)
2f54bf6e  =>  isOwner(address)
fe9fbb80  =>  isAuthorized(address)
f2fde38b  =>  transferOwnership(address)
c9c65396  =>  createPair(address,address)
fff6cae9  =>  sync()
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,add
ress,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
5c11d795  =>  swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,ui
nt256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,addre
ss[],address,uint256)
791ac947  =>  swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint2
56,address[],address,uint256)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
d0e30db0  =>  deposit()
ffb2c479  =>  process(uint256)
8c21cd52  =>  shouldDistribute(address)
5319504a  =>  distributeDividend(address)
f0fc6bca  =>  claimDividend()
28fd3198  =>  getUnpaidEarnings(address)
e68af3ac  =>  getCumulativeDividends(uint256)
db29fe12  =>  addShareholder(address)
9babdad6  =>  removeShareholder(address)
93028afd  =>  rebase_percentage(uint256,bool)
7a43e23f  =>  rebase(uint256,int256)
571ac8b0  =>  approveMax(address)
cb712535  =>  _transferFrom(address,address,uint256)
f0774e71  =>  _basicTransfer(address,address,uint256)
4afa518a  =>  checkTxLimit(address,uint256)
e7c44c69  =>  shouldTakeFee(address)
a7dd4bbc  =>  takeFee(address,uint256,bool)
0d5c6cea  =>  shouldSwapBack()
1da1db5e  =>  clearStuckBalance(uint256)
44a33fd2  =>  clearStuckBalance_sender(uint256)
ec72d65f  =>  set_sell_multiplier(uint256)
26e353b8  =>  tradingStatus(bool,uint256)
9ba1fc4c  =>  launchStatus(uint256)
ff7da74e  =>  enable_hotel_CaliforniaMode(bool)
fe2840e4  =>  set_max_roomrent(uint256)
ca1d908c  =>  manage_houseguests(address[],bool)
2d594567  =>  cooldownEnabled(bool,uint8)
6ac5eeee  =>  swapBack()
```

```
f708a64f  =>  setIsDividendExempt(address,bool)
658d4b7f  =>  setIsFeeExempt(address,bool)
f84ba65d  =>  setIsTxLimitExempt(address,bool)
50db71fb  =>  setIsTimelockExempt(address,bool)
04a66b48  =>  setFees(uint256,uint256,uint256,uint256,uint256)
d7c01032  =>  setFeeReceivers(address,address,address)
df20fd49  =>  setSwapBackSettings(bool,uint256)
201e7991  =>  setTargetLiquidity(uint256,uint256)
753d02a1  =>  manualSync()
2f34d282  =>  setLP(address)
26fae0d3  =>  setMaster(address)
83b4ac68  =>  isNotInSwap()
6d351d1a  =>  checkSwapThreshold()
9d1944f5  =>  setDistributorSettings(uint256)
33f3d628  =>  rescueToken(address,uint256)
2b112e49  =>  getCirculatingSupply()
d51ed1c8  =>  getLiquidityBacking(uint256)
1161ae39  =>  isOverLiquified(uint256,uint256)
b43b7835  =>  checkMaxWalletToken()
6149a20a  =>  checkMaxTxAmount()
09302dc6  =>  setMaxWalletPercent_base1000(uint256)
bd9ab537  =>  setMaxTxPercent_base1000(uint256)
1ca0a28d  =>  multiTransfer(address,address[],uint256[])
335f6a43  =>  multiTransfer_fixed(address,address[],uint256)
```

SWC Attacks

Part of the software analysis is the check for smart contract vulnerabilities. These vulnerabilities are registered at the SWC Registry which is an implementation of the weakness classification scheme proposed in EIP-1470. The goals of the SWC Registry are:

- Providing a straightforward way to classify security issues in smart contract systems
- Define a common language for describing security issues in smart contract systems' architecture, design, or code
- Serve as a way to train and increase performance for smart contract security analysis tools

Token Raven uses Consensys software tools to analyze smart contracts on SWC attacks.

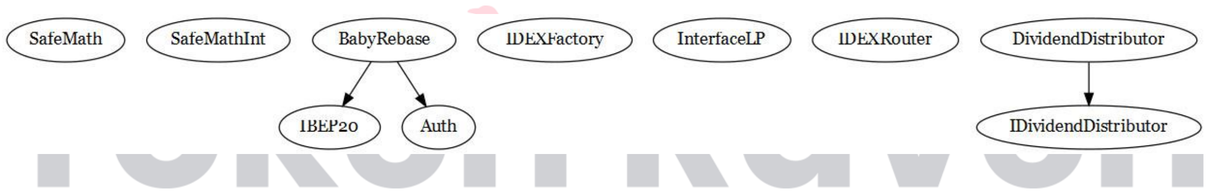
ID	Title	Status
SWC-136	Unencrypted Private Data On-Chain	Passed
SWC-135	Code With No Effects	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-131	Presence of unused variables	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-129	Typographical Error	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-123	Requirement Violation	Passed
SWC-122	Lack of Proper Signature Verification	Passed

SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-119	Shadowing State Variables	Passed
SWC-118	Incorrect Constructor Name	Passed
SWC-117	Signature Malleability	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-113	DoS with Failed Call	Passed
SWC-112	Delegatecall to Untrusted Callee	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-110	Assert Violation	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-107	Reentrancy	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-105	Unprotected Ether Withdrawal	Passed

SWC-104	Unchecked Call Return Value	Passed
SWC-103	Floating Pragma	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-100	Function Default Visibility	Passed

Inheritance graph

Inheritance is the procedure in which one class inherits the attributes and methods of another class. The class whose properties and methods are inherited is known as the Parent class.



Call Graph

A call graph is a control-flow graph which represents calling relationships between subroutines in a smart contract. Each node represents a procedure, and each edge (f, g) indicates that procedure f call procedure g.

Link:

https://github.com/TokenRaven/Audits/blob/main/BabyRebase/babyrebase%20call%20graph.png

Manual Analysis

Our manual analysis is focused on attributes that can not be verified or assessed through software analysis.

Owner Privileges

Owner can set fees higher than 25%

Owner can exclude wallets from fee. Owner can exclude wallet from reflections.

```
function setFees(uint256 _liquidityFee, uint256 _reflectionFee, uint256
_marketingFee, uint256 _devFee, uint256 _feeDenominator) external authorized {
        liquidityFee = _liquidityFee;
        reflectionFee = _reflectionFee;
        marketingFee = _marketingFee;
        devFee = _devFee;
        totalFee =
_liquidityFee.add(_reflectionFee).add(_marketingFee).add(_devFee);
        feeDenominator = _feeDenominator;
        require(totalFee < feeDenominator/3, "Fees cannot be more than 33%");
    }
```

Recommendation:

We advise the project owner to lower the maximum fee settings to at most 25%. This to protect the holders and investors.

Owner can exclude wallets from rewards and fees

Owner can exclude wallets from fee. Owner can exclude wallet from dividend.

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

Recommendation:

We recommend the project owner to be fully transparent about which wallets are excluded from fees. We also recommend the project owner to be fully transparent about which wallets are excluded from dividends. Or even better, remove this feature from the contract. And otherwise conduct an official KYC at Token Raven to gain trust of the community.

Owner can disable trading

Owner can disable trading.

```
function tradingStatus(bool _status, uint256 _deadBlocks) public onlyOwner {
        tradingOpen = _status;
        if(tradingOpen && launchedAt == 0){
            launchedAt = block.number;
            deadBlocks = _deadBlocks;
        }
    }
```

Recommendation:

We recommend finding another way to start trading after launch as this function allows the owner not only to open trading after launch but also disable it.

Use of block.timestamp

Contract makes use of block.timestamp, timestamp can be manipulated by miners.

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

Recommendation:

We recommend to avoid relying on the block timestamp.

Issue	**Category**	**Risk**	**Status**
Owner can set tax fee percentage higher than 25%	Owner privileges	Medium	Unresolved
Owner can exclude wallets from rewards and fees	Owner privileges	Low	Unresolved
Use of block.timestamp	Use of block.timestamp	Low	Unresolved
Owner can disable trading	Disable trading	Medium	Unresolved

Risk Summary

Vulnerability level	**Number of Issues**	**Issues resolved**
🔴 High-risk	0	0
🟡 Medium-risk	2	0
🟢 Low-risk	2	0

Conclusion

Token Raven has performed a combination of manual and software analysis of the smart contracts provided by the project team. Each of them has been analyzed for the most common and exploits, vulnerabilities, and manipulation attacks.

For Baby Rebase we have come to the following conclusion.

- The solidity source code has **Low Severity**
- The smart contract has an **Active Ownership**
- The centralization risk is therefore **high**
- The only Owner privileges are **changing fees, exclude wallets from fee and disable trading etc.**

Disclaimer

The audits provided by Token Raven provide projects with a comprehensive analysis regarding all possible vulnerabilities in their smart contracts. This includes for example, warnings, typos, unused functions & variables, compiler version, centralization check and any other hazard. As such our report and recommendations can be used to remedy vulnerabilities.

The Token Raven audit service is sure to improve the trustworthiness and safety of any legitimate project.

However, all the content provided in the Token Raven audit reports are for general information and should not be used as financial advice nor a reason to invest in a project. The Token Raven team is not responsible in the event of a fraud or scam committed by any of our partners. We implore everyone to always do your own due diligence.

In the event of a scam

If a project that we have KYC verified commits illegal or scam activities, it will be reported to the proper authorities (i.e., law enforcement etc.) After reporting the scam, we will conduct a thorough investigation into its activities and may release the KYC information to the public and on our website if it turns out that the reported project is confirmed to be a scam. This to prevent individuals with malicious intent from committing further frauds in the future.

To report a scam project, you can contact us on telegram and provide us with the following information:

- Presale link (if available)
- Smart contract address
- Proof of scam activities
- Audit and KYC certificate

About

We provide your blockchain project with KYC, Audit & Consultancy services on a daily timeframe. Thus far we have assisted over 50 projects to launch successfully.

Why Token Raven?

The cold hard logic of smart contracts allows them to facilitate the most secure and trustworthy kinds of transactions to date. This infallible ability to always execute the provided code does mean however, that the code should be free of errors to prevent security hazards.

TR takes pride in the fact that our audits stand for a secure smart contract. Not only do we provide a seal of approval, before publishing any audit we also provide developer teams with detailed insights as to how they can resolve potential security issues. The track record at Token Raven means that our audits provide legitimacy and security to any legitimate project.