PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For BabyCrib

22 October 2021

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

1 Overview

This report has been prepared for BabyCrib on the Binance Smart Chain (BSC). Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	BabyCrib
URL	TBC
Platform	Binance Smart Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`DividendDistributor`	`0x94acddffc98c9e5b27ccede77c89346b63ec9e58`	MATCH
`BabyCrib`	`0xD1FcB3A42E91e1847d43E28a129e8F31a415d3CD`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	6	1	-	5
Medium	11	4	-	7
Low	5	3	-	2
Informational	12	5	1	6
Total	**34**	**13**	**1**	**20**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to 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.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 DividendDistributor

ID	Severity	Summary	Status
01	HIGH	Client has completely replaced the initial contract with an unaudited one	ACKNOWLEDGED
02	HIGH	The contract can self-destruct, making BabyCrib completely unusable and potentially allowing for the deployer to deploy a new, malicious contract at this specific address using deterministic deployment	ACKNOWLEDGED
03	HIGH	BABY and WBNB addresses can be swapped out	ACKNOWLEDGED
04	MEDIUM	Distribution criteria lacks safeguards	ACKNOWLEDGED
05	MEDIUM	process and processManually may run out of gas	ACKNOWLEDGED
06	MEDIUM	distributeDividends may fail if set to a contract without the payable modifier	RESOLVED
07	LOW	claimDividend is not strictly secure from reentrancy	RESOLVED
08	INFO	Repeatedly calling removeShareholder would result in other users being removed	RESOLVED
09	INFO	Contract does not require SafeMath library anymore and can save on gas	ACKNOWLEDGED
10	INFO	Lack of events for setDistributionCriteria, setShare, deposit and distributeDividend	PARTIAL

1.4.2 BabyCrib

ID	Severity	Summary	Status
11	HIGH	swapBack functionality mints new tokens to the marketing address instead of using the balance of the contract which allows marketing to potentially mint and dump large amounts of tokens	ACKNOWLEDGED
12	HIGH	`setRouterAddress` could be used to turn the token into a honeypot	ACKNOWLEDGED
13	HIGH	BABY and WBNB addresses can be swapped out	RESOLVED
14	MEDIUM	`includeInReward` causes a previously excluded address to gain reflection rewards at the expense of other addresses' reflection fees	RESOLVED
15	MEDIUM	Governance can swap out the distributor contract for one that potentially reverts transactions or steals all reflection fees	ACKNOWLEDGED
16	MEDIUM	`transferBNBToAddress` may fail if `marketingAddress` is to a contract without the `payable` modifier	RESOLVED
17	MEDIUM	Setting tax fee to zero is not possible if `taxFee` was previously set to a positive number	RESOLVED
18	MEDIUM	Total fees is too high at 50%	ACKNOWLEDGED
19	MEDIUM	There are no safeguards for the anti-whale, `setTokenSwapThreshold` and `setBuybackBNBThreshold`	ACKNOWLEDGED
20	MEDIUM	`getRate` and `getCurrentSupply` may run out of gas	ACKNOWLEDGED
21	MEDIUM	Owner can exclude any address from receiving dividends	ACKNOWLEDGED
22	LOW	Owner can transfer all BNB and non-BabyCrib tokens	ACKNOWLEDGED
23	LOW	reflectBabyFee is unused	ACKNOWLEDGED
24	LOW	Liquidity tokens are sent to the owner account	RESOLVED
25	LOW	`manuallyProcessDividends` is not strictly secure from reentrancy	RESOLVED
26	INFO	Incorrect comments	RESOLVED
27	INFO	`dead` can be made constant	RESOLVED
28	INFO	Adding time limits to the router-related functions is unnecessary	ACKNOWLEDGED
29	INFO	`manuallyProcessDividends`, `reflect`, `withdrawForeignToken` and `withdrawBNB` functions can be made `external`	RESOLVED

30	INFO	`withdrawBNB`, `withdrawForeignToken`, `excludeFromFee`, `includeInFee`, `setMaxTxAmount`, `setTokenSwapThreshold`, `setMarketingAddress`, `setBuyback`, `setBuybackBNBThreshold`, `setBuybackUpperLimit`, `setFees`, `setLiquidity`, `excludeFromReward`, `includeInReward`, `manuallyProcessDividends`, `setDistributionCriteria`, `setDistributorGas` and `setIsDividendExcluded` does not emit an event	RESOLVED
31	INFO	_dexRouter variable is unecessary	ACKNOWLEDGED
32	INFO	Contract doesn't require SafeMath library anymore and can save on gas	ACKNOWLEDGED
33	INFO	OwnerWithdraw event in withdrawBNB emits as if WBNB was withdrawn while in fact BNB was withdrawn	ACKNOWLEDGED
34	INFO	Lack of safeTransfer usage	ACKNOWLEDGED

2 Findings

2.1 DividendDistributor

The DividendDistributor contract takes in BNB and performs a swap for BabySwap tokens, which represent the dividends that will be rewarded to users.

Note that the BNB swap amount and distribution period may be subject to change by the owner of the token contract.

2.2.1 Privileged Roles and actions

The following functions can be called by the owner of the contract:

- `pairToken`
- `transferTokenOwnership`
- `setMainTokenAddress`
- `upgradeDistributor`
- `setDistributionCriteria`
- `setShare`

2.2.2 Issues & Recommendations

Issue #01	**Client has completely replaced the initial contract with an unaudited one**
Severity	HIGH SEVERITY
Description	The client has replaced the original dividendDistributor with a distributor that contains completely novel code and has a fundamentally different logic in the sense that it distributes different tokens and in a different manner. Of the 199 original lines of code, 134 have been removed and 271 have been added. As our audits do not cover severe scope adjustments like this and the client wanted to launch within a very short timeframe that would not allow us to re-audit a whole new contract this new code could only be validated to a limited extent. The BabyCrib contract was also almost completely replaced but Paladin did take initiative to still include some of the issues we found on the new code to inform investors of potential risks and shortcomings within the new BabyCrib code. The initial issues have also been marked as resolved or acknowledged according to the new codebase.
Recommendation	Consider getting the new contract either properly peer-reviewed or audited.
Resolution	ACKNOWLEDGED

Issue #02	**The contract can self-destruct, making BabyCrib completely unusable and potentially allowing for the deployer to deploy a new, malicious contract at this specific address using deterministic deployment**
Severity	HIGH SEVERITY
Description	Governance can cause the DividendDistributor to selfdestruct, which means removing the contract completely from the blockchain. Adding this functionality to a contract is considered extremely bad practice as it allows the deployer, if it is a contract, to potentially replace this contract with a malicous one. Furthermore it would cause all babycrib functionality to stop working.
Recommendation	Consider removing the upgradeDistributor function.
Resolution	ACKNOWLEDGED

Issue #03	**BABY and WBNB addresses can be swapped out**
Severity	HIGH SEVERITY
Description	The `swapBabySwapAddress` allows the owner of the token contract to swap out the BABY token contract addresses in this Distributor contract, which would result in users receiving non-Babyswap tokens. The `setWBNBAddress` allows for the WBNB token contract address to be swapped out, though its effects would be limited since `swapExactETHForTokensSupportingFeeOnTransferTokens` uses BNB instead of WBNB. Nonetheless, we see no reason for this function to remain in the contract.
Recommendation	Consider removing both functions to better safeguard your users.
Resolution	ACKNOWLEDGED Although the WBNB token is no longer present, the baby token can still be swapped out.

Issue #04	Distribution criteria lacks safeguards
Severity	MEDIUM SEVERITY
Description	The `setDistributionCriteria` function allows for the token owner to specify the dividend distribution period, dividend amount and BNB swap amount. Unfortunately, these do not have any safeguards and can thus be set to any amount. Should the `minPeriod` be set to an unreasonably high figure (say 1 year), then dividend distributions would only occur in a year – which in our opinion is unreasonably long. The same can be extended to setting `minDistribution` to an unreasonably high figure. If it was, for example, set to 10 million tokens, then it would all but impossible for `shouldDistribute` to ever return `True` and thus users would not receive any dividends.
Recommendation	Consider setting reasonable upper and lower limits for `minPeriod`, `minDistribution` and `swapToBabySwapThreshold`.
Resolution	ACKNOWLEDGED

Issue #05	`process` may run out of gas
Severity	MEDIUM SEVERITY
Description	In the event that there are a very large number of shareholders, then the `process` function may run out of gas before all shareholders receive their dividends. Additionally, if an insufficient amount of gas is passed in, then both functions may again run out of gas before the full list of shareholders are paid.
Recommendation	Consider using mapping instead of arrays, and ensuring that a sufficiently high amount of gas is passed in to the functions.
Resolution	ACKNOWLEDGED

Issue #06	**`distributeDividends` may fail if set to a contract without the payable modifier**
Severity	MEDIUM SEVERITY

Location	Lines 618-629
	`function distributeDividend(address shareholder) internal {`
	`  if(shares[shareholder].amount == 0){ return; }`
	
	`  uint256 amount = getUnpaidEarnings(shareholder);`
	`  if(amount > 0){`
	`  totalDistributed = totalDistributed.add(amount);`
	`  IERC20(BABY).transfer(shareholder, amount);`
	`  shareholderClaims[shareholder] = block.timestamp;`
	`  shares[shareholder].totalRealised = shares[shareholder].totalRealised.add(amount);`
	`  shares[shareholder].totalExcluded = getCumulativeDividends(shares[shareholder].amount);`
	`  }`
	`}`

Description	Should the shareholder address be set to a contract that does not contain the payable modifier, then it would not be able to receive any BNB.
Recommendation	The simplest solution would be to use send instead of `transfer`. A better solution would be to instead use wBNB and `safeTransfer`, like so: `wBNB.safeTransfer(shareholder, amount);`
Resolution	RESOLVED The client has sent us completely new code which no longer distributes BNB dividends.

Issue #07	`claimDividend` is not strictly secure from reentrancy
Severity	LOW SEVERITY
Description	The `claimDividend` function may be susceptible to reentrancy, especially if ERC777 tokens are used. This may result in the contract being inadvertently exploitable, as was the case with the AMP token in Cream Finance just recently.
Recommendation	Consider adding in the `nonReentrant` modifier to the `claimDividend` function. Additionally, restricting this function to only be callable by non-contracts may add an additional layer of security (though any contracts would not be able to call this function anymore), like so: `bool isEOA = tx.origin == msg.sender && ! msg.sender.isContract();`
Resolution	RESOLVED The dividend claims now have reentrancy modifiers.

Issue #08	Repeatedly calling `removeShareholder` would result in other users being removed
Severity	INFORMATIONAL
Description	In the `removeShareholder` function, the removed account's index position is not zeroed out. This will result in other users being removed if this function is repeatedly called with the previously-removed user's address. The following situation demonstrates this issue: 1. `removeShareholder` is called to remove shareholder A. 2. Shareholder B now takes the position of shareholder A in the array. 3. Calling `removeShareholder` again with shareholder A's address would now remove shareholder B.
Recommendation	Consider setting the `shareholderIndexes` of the removed account to 0.
Resolution	RESOLVED

Issue #09	**Contract does not require SafeMath library anymore and can save on gas**
Severity	INFORMATIONAL
Description	The contract was deployed on Solidity version 0.8.7, which implements all SafeMath checks by default. As such, all arithmetic operators can simply use their raw operators such as + or - to save on gas.
Recommendation	Feel free to use the raw arithmetic operators through the contract to marginally save on gas costs.
Resolution	ACKNOWLEDGED

Issue #10	**Lack of events for `setDistributionCriteria`, `setShare`, `deposit` and `distributeDividend`**
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	PARTIALLY RESOLVED Some of the functions have events now.

2.2 BabyCrib

The BabyCrib contract provides both reflection and dividend income to users, with the operations of the latter mainly handled by the DividendDistributor that is provided during the deployment of this contract and can be changed by the governance. The owner of the contract has the ability to exclude or include certain addresses from receiving rewards, and there are several fees associated. It should be noted that the owner can exclude address from receiving rewards, which should be done sparingly if at all.

The owner can withdraw BNB and non-BabyCrib tokens, which should not pose an issue as users should not have any funds at risk in the contract. Finally, a 1% anti-whale has been implemented which limits transfer sizes. This anti-whale variable can be set artificially low and is not present on purchases which is dangerous as it could prevent people from selling their tokens again. The contract furthermore contains a large array of governance privileges which could be considered excessive if the owner address is untrusted or not properly locked down.

2.2.1 Token Overview

Address	0xd1fcb3a42e91e1847d43e28a129e8f31a415d3cd
Token Supply	1,000,000,000 (one billion) - all pre-minted but bug allows governance to mint more
Decimal Places	18
Transfer Max Size	1%
Transfer Min Size	None
Transfer Fees	Variable depending on transfer type (maximum 50%)

2.2.2 Privileged Roles and Actions

The following functions can be called by the owner of the contract:

- `transferOwnership`
- `withdrawBNB`
- `withdrawForeignToken`
- `setRouterAddress`
- `setPairAddress`
- `setIsLiquidityPool`
- `excludeFromRewards`
- `setFeeExemption`
- `setTxLimitExempt`
- `setMaxTxAmount`
- `upgradeDistributor`
- `setTokenSwapThreshold`
- `setMarketingAddress`
- `setFees`
- `includeInRewards`
- `setDistributorGas`

2.1.2 Issues & Recommendations

Issue #11	**swapBack functionality mints new tokens to the marketing address instead of using the balance of the contract which allows marketing to potentially mint and dump large amounts of tokens**
Severity	HIGH SEVERITY
Description	The `swapBack` function is supposed to take tokens from the contract address and send them to the marketing wallet. Instead, it uses the `_sendTokens` function to simply increase the reflection amount of the marketing wallet without increasing the visible supply of the token. This means that the contract is minting tokens out of thin air without increasing the supply. This furthermore means that this functionality could potentially be abused by the marketing wallet to mint huge amounts of tokens to themselves to drain the liquidity of all funds. The fact that the governance has such powers might be bad for investor confidence, this issue is marked as high risk because of this reason but also because the business logic of the function is fundamentally wrong by not using the balance inside the contract.
Recommendation	Consider actually subtracting the `t/rAmount` from the token contract.
Resolution	ACKNOWLEDGED

Issue #12	`setRouterAddress` could be used to turn the token into a honeypot
Severity	HIGH SEVERITY
Description	The upgraded router contract could be a malicious contract that simply keeps the tokens sent to it, or even worse a contract that prevents selling transactions by always reverting.
Recommendation	Consider removing this function. It is very rare that any protocol should ever require changing the router and generating a new LP pair address.
Resolution	ACKNOWLEDGED The new version of the code has a function to change the pair and potentially allow including the pair again which could all be used for similar malice in case the owner would turn malicious or keys are compromised.

Issue #13	BABY and WBNB addresses can be swapped out
Severity	HIGH SEVERITY
Description	Note that this issue has been listed in the DividendDistributor audit above, though we have nonetheless included it here as a reminder that these functions should be removed. The `swapBabySwapAddress` allows the owner of the token contract to swap out the BABY token contract addresses in this Distributor contract, which would result in users receiving non-Babyswap tokens. The `setWBNBAddress` allows for the WBNB token contract address to be swapped out, though its effects would be limited since `swapExactETHForTokensSupportingFeeOnTransferTokens` uses BNB instead of WBNB. Nonetheless, we see no reason for this function to remain in the contract.
Recommendation	Consider removing both functions to better safeguard your users.
Resolution	RESOLVED Both functions have been removed.

Issue #14	**`includeInReward` causes a previously excluded address to gain reflection rewards at the expense of other addresses' reflection fees**
Severity	MEDIUM SEVERITY
Description	As described in this blog article, a previously excluded address would gain some reflection fees at the expense of other addresses' reflection fees. This could cause a loss of some balances of users. This is done because `_rOwned` is not updated in `includeInFee`. This function is only callable by the contract's owner.
Recommendation	Consider never including a previously excluded address, as there is currently no reasonably effective method to solve this issue.
Resolution	RESOLVED The client has included logic that decreases rTotal with the reflected amount for this user. The user furthermore does not earn this reflected amount when they are included again. It should be noted that the safemoon way of handling exclusions is always fundamentally lacking in its design and that this issue is more a hotfix than a fundamental fix of the problem which is that the exclusion mechanism should not retain the rAmount of the excluded accounts.

Issue #15	**Governance can swap out the distributor contract for one that potentially reverts transactions or steals all reflection fees**
Severity	MEDIUM SEVERITY
Description	The contract allows the governance to upgrade the distributor contract to a new version which could allow them to upgrade this contract to a malicious version to potentially steal reflection fees or do other malice like reverting specific transactions
Recommendation	Consider not making the distributor upgradeable.
Resolution	ACKNOWLEDGED

Issue #16	**`transferBNBToAddress` may fail if `marketingAddress` is to a contract without the `payable` modifier**
Severity	MEDIUM SEVERITY

Location	Lines 806-808
	`function transferBNBToAddress(address payable recipient, uint256 amount) private {`
	`    recipient.transfer(amount);`
	`}`

Description	Should the `marketingAddress` address be set to a contract that does not contain the payable modifier, then it would not be able to receive any BNB.
Recommendation	The simplest solution would be to use send instead of transfer. A better solution would be to instead use wBNB and safeTransfer, like so: `WBNB.safeTransfer(recipient, amount);`
Resolution	RESOLVED The new code the client sent to us is very different but no longer reverts if the marketing address is set to zero.

Issue #17	Setting tax fee to zero is not possible if `taxFee` was previously set to a positive number
Severity	MEDIUM SEVERITY
Location	Lines 980-992

```
function removeAllFees() private {
    if(_taxFee == 0) return;

    _previousTaxFee = _taxFee;
    _taxFee = 0;
}

/**
 * @notice Restores the fees
 */
function restoreAllFees() private {
    _taxFee = _previousTaxFee;
}
```

Lines 1126-1137

```
// Remove fees completely from the transfer if either wallet are
excluded
if (!takeFee) {
    removeAllFees();
}

// Transfer the token amount from sender to receipient.
_tokenTransfer(from, to, amount);

// If we removed the fees for this transaction, then restore them
for future transactions
if (!takeFee) {
    restoreAllFees();
}
```

Description

Assuming that:

- `previousTaxFee` is 30%
- `takeFee == false`
- `setFees` has been called to set `_taxFee` to 0

When the `_transfer` function is called, `removeAllFee` would be triggered. The following series of events would thus occur:

1. Since `taxFee == 0` is `True`, `removeAllFee` would return early. `previousTaxFee` would still remain as 30%.
2. `_tokenTransfer` would pass without issue.
3. `restoreAllFee` would then set `_taxFee` to `_previousTaxFee`.

The end result of this is that the current `taxFee` is now 30% again.

Recommendation

Consider setting both previous and current tax fees to the new figure if setFees is called, like so:

```
function setFees(uint256 liquidityFee, uint256 buybackFee, uint256
reflectFee, uint256 reflectbabyFee, uint256 marketingFee) external
onlyOwner {
    _liquidityFee = liquidityFee;
    _buybackFee = buybackFee;
    _reflectFee = reflectFee;
    _reflectbabyFee = reflectbabyFee;
    _marketingFee = marketingFee;
    _taxFee =
_liquidityFee.add(_buybackFee).add(_reflectFee).add(_reflectbabyFe
e).add(_marketingFee);
    require(_taxFee < 50);

    _previousTaxFee = _taxFee;

    emit TaxUpdated(_taxFee);
}
```

Alternatively, if `_taxFee == 0`, then returning early in `restoreAllFee` could also be considered (similar to the mechanism in `removeAllFee`).

Resolution

RESOLVED

The remove and restore functionality has been removed.

Issue #18	Total fees is too high at 50%
Severity	MEDIUM SEVERITY
Description	The `setFees` function allows for the various fees to be set to any amount, so long as the total `_taxFee` is capped at 50%. Should this upper limit be reached, almost half of any user's transfers would be deducted as fees (before accounting for some reflection and dividends), which in our opinion is too high. Even the current 30% can be considered too high by most standards.
Recommendation	Consider limiting `_taxFee` to at most 10%, which is much more reasonable and acceptable by most users.
Resolution	ACKNOWLEDGED Within the new contract the total amount of fee levied can even exceed this 49% maximum (as 50% is exclusive) causing the fee logic to revert and transfers to fail.

Issue #19	**There are no safeguards for the anti-whale, `setTokenSwapThreshold` and `setBuybackBNBThreshold`**
Severity	MEDIUM SEVERITY
Description	The anti-whale limit can be set arbitrarily low to 0.01% of the total supply. If the governance ever decides to do this, it would make it difficult or nearly impossible to transfer or sell tokens, essentially turning it into a honeypot, if `_maxTxAmount` is set to 0. Should the `tokenSwapThreshold` be set to a too high figure, then `swapAndLiquify` would not trigger, thus affecting liquidity, dividends distribution and all fee-related functionalities. If it is set too low, then `swapAndLiquify` would occur too often, potentially racking up large cumulative gas costs over time. Finally, if `buybackBNBThreshold` is set too high, then the `buybackTokens` function would not be called, or not called often enough to buy back and burn BabyCrib tokens.
Recommendation	Consider adding a lower limit to the `setMaxTxAmount` of for example 1-2% of the total supply. This would come in the form of adding the following line: `require(maxTxAmount >= 20000000 ether, "Too low anti-whale!");` Consider also setting reasonable limits to the `setBuybackBNBThreshold` and `setTokenSwapThreshold` functions.
Resolution	ACKNOWLEDGED `setBuybackBNBThreshold` has been removed however.

Issue #20	`getRate` and `getCurrentSupply` may run out of gas
Severity	MEDIUM SEVERITY
Description	In the event that there are a very large number of excluded address, the `getRate` and `getCurrentSupply` functions may run out of gas before the functions can complete their iterations.
Recommendation	Consider using mapping instead of arrays.
Resolution	ACKNOWLEDGED

Issue #21	Owner can exclude any address from receiving dividends
Severity	MEDIUM SEVERITY
Description	By calling the `setIsDividendExcluded`, the owner can exclude any address from receiving dividends.
Recommendation	Consider setting the ownership of the contract behind a sufficiently long Timelock, ideally 1 day. This would allow users to monitor the Timelock and react accordingly should this function be queued.
Resolution	ACKNOWLEDGED

Issue #22	Owner can transfer all BNB and non-BabyCrib tokens
Severity	LOW SEVERITY

Location	Lines 790-801
	`function withdrawBNB(uint256 amount) public onlyOwner() {`
	`    if(amount == 0)`
	`payable(owner()).transfer(address(this).balance);`
	`    else payable(owner()).transfer(amount);`
	`}`
	
	`/**`
	` * @notice Withdraws non-CRIB tokens that are stuck as to not interfere with the liquidity`
	` */`
	`function withdrawForeignToken(address token) public onlyOwner() {`
	`    require(address(this) != address(token), "Cannot withdraw native token");`
	`    IERC20(address(token)).transfer(msg.sender, IERC20(token).balanceOf(address(this)));`
	`}`

Description	These functions allow the owner to withdraw native BNB and all tokens sent to this contract by mistake, except for BabyCrib. This issue is marked as Low Risk because there should not be any user funds that can be transferred out using these functions, though it would affect the buyback mechanism if all BNB is continually taken out.
Recommendation	Consider placing the ownership of the contract behind a sufficiently long Timelock. Additionally, consider using `safeTransfer` instead of `transfer`.
Resolution	ACKNOWLEDGED

Issue #23	reflectBabyFee is unused
Severity	LOW SEVERITY
Description	The contract contains a settable reflectBabyFee which is unused.
Recommendation	Consider removing reflectBabyFee.
Resolution	ACKNOWLEDGED

Issue #24	Liquidity tokens are sent to the owner account
Severity	LOW SEVERITY
Description	Part of the transfer fees are used to generate liquidity. Currently, these LP tokens are being sent directly to the owner account, who would be able to dump them at any time.
Recommendation	Consider locking the liquidity forever in the contract, burning the tokens, or adding a simple vesting contract that gradually releases the LP tokens over time.
Resolution	RESOLVED There is no more liquidity fee.

Issue #25	`manuallyProcessDividends` is not strictly secure from reentrancy
Severity	LOW SEVERITY
Description	The `manuallyProcessDividends` function may be susceptible to reentrancy, especially if ERC777 tokens are used. This may result in the contract being inadvertently exploitable, as was the case with the AMP token in Cream Finance just recently.
Recommendation	Consider adding in the `nonReentrant` modifier to the `manuallyProcessDividends` function. Additionally, restricting this function to only be callable by non-contracts may add an additional layer of security (though any contracts would not be able to call this function anymore), like so: `bool isEOA = tx.origin == msg.sender && ! msg.sender.isContract();`
Resolution	RESOLVED This function has been removed.

Issue #26	Incorrect comments
Severity	INFORMATIONAL
Description	Throughout the contract, comments reference the PancakeSwap router. However, the BabyCrib token uses the Baby router instead (0x325E343f1dE602396E256B67eFd1F61C3A6B38Bd). Additionally, the token limits comments are incorrect. The anti-whale has been set to 1/200ths of 1 billion, and the `tokenSwapThreshold` amount is 1/1000ths of a billion. The `setTaxFeePercent` function can be removed as it has been commented out. Finally, the `setDistributionCriteria` function takes in `bnbToSafemoonThreshold`, which we assume to be incorrectly named.
Recommendation	Consider correcting these errors throughout the contract, and deleting unused functions.
Resolution	RESOLVED

Issue #27	dead can be made constant
Severity	INFORMATIONAL
Description	Variables that are never changed after deployment can be set as constant. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `dead` explicitly constant.
Resolution	RESOLVED The variable has been renamed to `_burnWallet` and is now constant.

Issue #28	Adding time limits to the router-related functions is unnecessary
Severity	INFORMATIONAL
Description	Throughout the contract, the router-related functions add 300 seconds for the deadline. This is generally not required and the 300 seconds can just be removed.
Recommendation	Consider removing the 300 seconds such the deadlines is simply `block.timestamp`.
Resolution	ACKNOWLEDGED

Issue #29	`manuallyProcessDividends`, `reflect`, `withdrawForeignToken` and `withdrawBNB` functions can be made `external`
Severity	INFORMATIONAL
Description	The above functions can be changed from `public` to `external`. Apart from being a best practice when the function is not used within the contract, this can lead to a lower gas usage in certain cases.
Recommendation	Consider making these functions `external`.
Resolution	RESOLVED

Issue #30	**`withdrawBNB`, `withdrawForeignToken`, `excludeFromFee`, `includeInFee`, `setMaxTxAmount`, `setTokenSwapThreshold`, `setMarketingAddress`, `setBuyback`, `setBuybackBNBThreshold`, `setBuybackUpperLimit`, `setFees`, `setLiquidity`, `excludeFromReward`, `includeInReward`, `manuallyProcessDividends`, `setDistributionCriteria`, `setDistributorGas`, `setIsDividendExcluded` does not emit an event**
Severity	INFORMATIONAL
Description	Functions that affect sensitive state variables should emit an event.
Recommendation	Add events for the above functions.
Resolution	RESOLVED All functions which the client has retained in their final version now have events.

Issue #31	**_dexRouter variable is unecessary**
Severity	INFORMATIONAL
Description	The code initializes a _dexRouter variable which holds the initial value of the _router variable. However, it suffices to simply initialize _router with the initial variable directly.
Recommendation	Consider removing _dexRouter and initializing _router directly with the address.
Resolution	ACKNOWLEDGED

Issue #32	Contract doesn't require SafeMath library anymore and can save on gas
Severity	INFORMATIONAL
Description	The contract was deployed on Solidity version 0.8.7, which implements all SafeMath checks by default. As such, all arithmetic operators can simply use their raw operators such as + or - to save on gas.
Recommendation	Feel free to use the raw arithmetic operators through the contract to marginally save on gas costs.
Resolution	ACKNOWLEDGED

Issue #33	OwnerWithdraw event in withdrawBNB emits as if WBNB was withdrawn while in fact BNB was withdrawn
Severity	INFORMATIONAL
Description	The OwnerWithdraw event within the withdrawBNB function emits that WBNB was withdrawn while in fact the non-wrapped version was withdrawn.
Recommendation	Consider using a separate event or a separate encoding (eg. token 0x0) to indicate that BNB is withdrawn.
Resolution	ACKNOWLEDGED

Issue #34	Lack of safeTransfer usage
Severity	INFORMATIONAL
Description	The code uses raw token transfer in the withdrawBNB function to transfer tokens, this might not work with tokens which are not strictly compliant with the ERC-20 standard and that do not return a boolean on transfer. Such tokens will not be withdrawable and the withdrawBNB function will revert these transfers as the decoding of the return value fails.
Recommendation	Consider using safeTransfer from the SafeERC20 OpenZeppelin library.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY