PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For TorCash (BSC)

13 July 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 TorCash 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	TorCash
URL	https://torcash.io/
Platform	Binance Smart Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
Verifier	0x77b5cd924fb56a5c6690dda4bb6e0c67f8b34d87	✓ MATCH
Hasher*	0xb8941F853cCBcb4AcF0F8fe40B06E0Cc24202854	✓ MATCH
Anonymous Tree**	0x728b3880DC04c8C3319b6729Fd339b35e384a0B9	✓ MATCH
Masterchef	0x4c5Eb4ABF6CAF4acFe70E238554D9d284793f36d	✓ MATCH
Anonymous-tree Deploy	0x678aB8a6e336482cd6FC415de3eF99cf2EeA4CC4	✓ MATCH
TorCoin	0xa3b6618f932d6c6b5252a501da50e3069dd049c8	✓ MATCH
Timelock	0xda1e55da00265ca9ef2ddbefd7c5c49ab31eafa9	✓ MATCH

* Verified using compileHasher.js in github.com/tornadocash/tornado-core.
** There are 42 copies of the Anonymous Tree contract, and the listed contract above is the one we have reviewed.

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	10	7	2	1
Medium	4	1	1	2
Low	11	6	-	5
Informational	12	2	1	9
Total	**37**	**16**	**4**	**17**

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 Verifier

ID	Severity	Summary	Status
01	HIGH	Verifier not generated with secure randomization could result in loss of all funds	RESOLVED
02	MEDIUM	Verifier implementation differs from Tornado Cash implementation	RESOLVED

1.3.2 Hasher

ID	Severity	Summary	Status
03	INFORMATIONAL	Users will have to reverify the Hasher contract using circomlib if a redeployment will happen	RESOLVED

1.3.3 Anonymous Tree

ID	Severity	Summary	Status
04	MEDIUM	Anonymous Tree differs from the Tornado Cash specification	ACKNOWLEDGED

1.3.4 MasterChef

ID	Severity	Summary	Status
05	HIGH	the `_paths` parameter can changed to drain the masterchef of any token desired	RESOLVED
06	HIGH	`updateTor` can be called by owner to change the native token to any other token and drain the masterchef of any token desired	RESOLVED
07	HIGH	Denominations can be updated to steal all staked tokens	PARTIALLY RESOLVED
08	HIGH	`anonymousTrees` can be updated to steal all staked tokens	PARTIALLY RESOLVED
09	HIGH	Deposit fees up to 100% possible	RESOLVED
10	HIGH	`updateSwapRouter` could be used to siphon all swap fees or block deposits	RESOLVED
11	HIGH	Transfer tax subtracted after `swapForTors` is called can result in severe miscalculation of rewards and balances	ACKNOWLEDGED
12	MEDIUM	Lack of checks for array parameters may result in running out of gas	PARTIALLY RESOLVED
13	MEDIUM	Potential overflow in `withdraw` function	ACKNOWLEDGED
14	LOW	Depositing funds without binding a note can result in lost funds	ACKNOWLEDGED
15	LOW	Frontend can spy on secrets and when compromised could withdraw for any secret captured	ACKNOWLEDGED
16	LOW	`Initialize` might not update the `lastRewardBlock` correctly, causing pools to not start at the expected block	RESOLVED
17	LOW	Masterchef will not work for LP tokens	RESOLVED
18	LOW	When a pool is added, the `lastRewardBlock` can be set manually causing an exceptionally high mint if this is set to a block in the past	RESOLVED
19	LOW	No reentrancy guards on external functions	ACKNOWLEDGED
20	LOW	`getPoolInfo(uint256_pid)` can run out of gas	RESOLVED
21	INFORMATIONAL	`target.call.value` is deprecated	ACKNOWLEDGED
22	INFORMATIONAL	The `INITIALIZED` variable is private	ACKNOWLEDGED
23	INFORMATIONAL	`addPool` casts the `_lpToken` to `ITorCoin` instead of `IBEP20` (typo)	ACKNOWLEDGED
24	INFORMATIONAL	`findDenomination` may run out of gas	RESOLVED

25	INFORMATIONAL	`public` functions not used internally can be made `external`	ACKNOWLEDGED

1.3.5 Anonymous-tree Deploy

ID	Severity	Summary	Status
26	INFORMATIONAL	Verifier not generated with secure randomization could result in loss of all funds	PARTIALLY RESOLVED

1.3.6 TorCoin

ID	Severity	Summary	Status
27	HIGH	There is no minimum for `maxTransferAmount` (anti-whale).	RESOLVED
28	HIGH	`updateTorSwapRouter` could be used to siphon all liquidity fees or even turn the token into a honeypot	RESOLVED
29	LOW	All initial LP tokens are sent to the operator	RESOLVED
30	LOW	`_inInitLiquify` is private	ACKNOWLEDGED
31	LOW	Token contract could build up dust BNB due to `addLiquidity`	ACKNOWLEDGED
32	LOW	The developer is excluded from anti-whale	RESOLVED
33	INFORMATIONAL	`initLiquify` could be abused if it is not called early on	ACKNOWLEDGED
34	INFORMATIONAL	Exclude relevant contracts from anti-whale	ACKNOWLEDGED
35	INFORMATIONAL	`initLiquify` can be made `external`	ACKNOWLEDGED
36	INFORMATIONAL	`delegateBySig` can be frontrun and cause denial of service	ACKNOWLEDGED
37	INFORMATIONAL	`target.call.value` is deprecated	ACKNOWLEDGED

1.3.7 Timelock

No issues found.

2 Findings

Important preliminary disclaimer

The TorCash team tries to ambitiously combine the Tornado Cash protocol inside the masterchef. This effectively allows for deposited funds to generate yield until they are eventually withdrawn. Paladin was commissioned to do an audit to specifically focus on user safety and potential issues in the project logic.

During this audit, our auditing team found a large degree of severe issues as documented in this report. Our auditors have given their best effort to find all issues with the protocol but since they've found so many issues, they cannot guarantee in any way that there are not more.

We thus recommend the TorCash protocol to carefully reconsider and reimplement their current protocol from the ground up, instead of patching each and every issue separately. Patching each and every issue separately may not result in a secure protocol due to the high level of interplay between components.

In the redesign of the protocol, we strongly recommend a careful consideration of the possible governance risks of each and every function added. We also recommend to copy Tornado Cash directly without any changes, as this will make it easier for future auditors to assess the security parameters of the protocol. Finally, we recommend separating the logic in different contracts to reduce the level of concern of each individual contract. It should be considered that each pool could have its own contract.

Update 13 July 2021

The team has implemented fixes for the issues raised, though we would like to reiterate that we cannot say for certain that all possible issues have been identified due to the extremely complex and highly interdependent nature of these contracts.

2.1 Verifier

2.1.2 Issues & Recommendations

Issue #01	Verifier not generated with secure randomization could result in loss of all funds
Severity	HIGH SEVERITY
Description	Currently the verifier we were given has been precomputed. The verifier essentially evaluates the zk-snark proof at "specific random points". If these points are not generated randomly or potentially maliciously this could result in the whole verification scheme to be gamed by a malicious party.
Recommendation(s)	Consider an MPC (multi-party computation) party similar to the one Tornado Cash has done. https://tornado-cash.medium.com/the-biggest-trusted-setup-ceremony-in-the-world-3c6ab9c8fffa
Resolution	RESOLVED Torcash has moved their verifier implementation to the official Tornado Cash v2.1 release (https://github.com/tornadocash/tornado-core/releases/tag/v2.1). This implementation has been generated by one of the largest MPC parties in history making this specific component secure.

Issue #02	Verifier implementation differs from Tornado Cash implementation
Severity	MEDIUM SEVERITY
Description	The verifier implementation is different then the implementation of Tornado Cash. We have compared this against the latest commit 77af0c5 (https://github.com/tornadocash/tornado-core/commit/77af0c5bddfcf9d973efbc38278a249bb0173da3).
Recommendation(s)	Consider not adding any custom logic to the Tornado Cash implementation to reduce the risk of making mistakes in the side effects of these modifications.
Resolution	RESOLVED Torcash has moved their verifier implementation to the official Tornado Cash v2.1 release (https://github.com/tornadocash/tornado-core/releases/tag/v2.1). This implementation has been generated by one of the largest MPC parties in history making this specific component secure.

2.2 Hasher

The hasher contract is an essential component of the deposit and withdrawal mechanism. It is responsible for verifying the proof that you are actually allowed to withdraw a deposit. If this code is untrusted, the whole protocol security falls apart.

Paladin has confirmed that the hasher contract at the address we've been provided matches the output of the Tornado Cash bytecode generator.

2.2.1 Issues & Recommendations

Issue #04	**Users will have to reverify the Hasher contract using circomlib if a redeployment will happen**
Severity	INFORMATIONAL
Description	Due to the Hasher contract not being written in solidity but instead being compiled using circomlib, it cannot be verified on the explorers. Instead, it has to be verified using circomlib.
Recommendation(s)	Consider adding a page that explains how users and reviewers can confirm the validity of the Hasher contract. This is done through a manual bytecode comparison with the bytecode output generated from tornadocash/tornado-core compileHasher.js.
Resolution	RESOLVED The client has provided us the latest hasher implementation (0xb894...2854) that matches the bytecode we verified.

2.3 Anonymous Tree

2.3.1 Issues & Recommendations

Issue #03	Anonymous Tree differs from the Tornado Cash specification
Severity	MEDIUM SEVERITY
Description	The Anonymous Tree contract is not present in Tornado Cash and could thus result in unexpected side effects if not implemented carefully with a complete understanding of the Tornado Cash protocol. We have compared this against the latest commit 77af0c5 (https://github.com/tornadocash/tornado-core/commit/77af0c5bddfcf9d973efbc38278a249bb0173da3).
Recommendation(s)	Consider not adding any custom logic to the Tornado Cash implementation to reduce the risk of making mistakes in the side effects of these modifications.
Resolution	ACKNOWLEDGED

2.4 Masterchef

2.4.1 Privileged Roles

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

- `setExcludedFromLP`
- `updateTor`
- `updateSwapRouter`
- `set`
- `addPool`
- `initialize`
- `transferOwnership`
- `renounceOwnership`

2.4.2 Issues & Recommendations

Issue #05	the `_paths` parameter can changed to drain the masterchef of any token desired
Severity	HIGH SEVERITY
Description	The owner can add a new pool or update an existing pool with a malicious `_paths` parameter. If the path is set to for example [BUSD, fake token] and the owner owns all the LP tokens of this pair, the owner can drain BUSD on every swap that is made.
Recommendation(s)	Consider making the tor token, router and the path from `depositToken` to tor static.
Resolution	RESOLVED The ownership has been renounced while the `_paths` parameter indicates a valid path from the pool tokens to tor.

Issue #06	`updateTor` can be called by owner to change the native token to any other token and drain the masterchef of any token desired
Severity	HIGH SEVERITY
Description	The owner could call the `updateTor` function to update the native token to a different token. Since the tor token is used to distribute the rewards, the owner could create a private pool to drain the masterchef of any token.
Recommendation(s)	Consider removing the `updateTor` function.
Resolution	RESOLVED Since ownership is renounced, this function can no longer be called.

Issue #07	Denominations can be updated to steal all staked tokens
Severity	HIGH SEVERITY
Description	The owner could update a pool's denominations to steal all LP tokens. This is because the trees are indexed by the denomination id, and into the actual denomination.
Recommendation(s)	Consider removing the updating of denominations in the set function.
Resolution	PARTIALLY RESOLVED Ownership is renounced but denominations could have been different in the past. We recommend investors to carefully go over all ownership transactions.

Issue #08	`anonymousTrees` can be updated to steal all staked tokens
Severity	HIGH SEVERITY
Description	The owner could add or change anonymous trees to steal all LP tokens. As these are used for the external `unBind` function call in the `withdraw` function, adding an arbitrary `_anonymousTrees` contract for a pool would allow arbitrary withdrawal of that pool's deposited LP tokens. Also, as there is no check for if the LP token to be added exists in the masterchef, it is possible to add a duplicate token in a pool with a malicious `_anonymousTrees` to drain deposits.
Recommendation(s)	Consider fundamentally rethinking the design of the masterchef. One idea would be to have subcontracts for every [token, denomination] pair. Each of these contracts could have an associated `anonymousTree` that is not upgradeable. This way, the pools are isolated from each other and logic is simplified.
Resolution	PARTIALLY RESOLVED Ownership is renounced but anonymous could be (and were) different in the past. We recommend investors to carefully go over all ownership transactions.

Issue #09	Deposit fees up to 100% possible
Severity	HIGH SEVERITY
Description	Currently the masterchef allows for `depositFees` up to 100%, all though we expect that this will not be set to this level, it is possible and may cause users to mistrust the project.

Recommendation(s)	Consider updating the caps to a reasonable amount like 4%:
	`require(_depositFeeBP <= 400, "add: invalid deposit fee basis points");`
	`require(_depositFeeBP <= 400, "set: invalid deposit fee basis points");`

Resolution	RESOLVED Ownership has been renounced.

Issue #10	`updateSwapRouter` could be used to siphon all swap fees or block deposits
Severity	HIGH SEVERITY
Description	The owner can update the Uniswap router address that is responsible for converting the transfer fee into BNB for liquidity. When this router is changed to a malicious contract, this could be used to block sell transactions (turning it into a honeypot). It could also be used to simply keep the transaction fees and send them to the owner.
Recommendation(s)	Consider removing this function. If this is not possible, consider using an "operator" account which is behind a significantly longer timelock so investors can reasonably see this change coming and inspect the new router.
Resolution	RESOLVED Ownership has been renounced and router is PancakeSwap v2.

Issue #11	Transfer tax subtracted after `swapForTors` is called can result in severe miscalculation of rewards and balances
Severity	HIGH SEVERITY
Description	Due to the transfer tax being deducted from `_denomination` instead of `_amount`, it is being based on an amount smaller than the actual amount that was transferred in.
Recommendation(s)	Consider removing this function. If this is not possible, consider using an "operator" account which is behind a significantly longer timelock so investors can reasonably see this change coming and inspect the new router.
Resolution	ACKNOWLEDGED The tor pool is vulnerable.

Issue #12	Lack of checks for array parameters may result in running out of gas
Severity	MEDIUM SEVERITY
Description	The `poolInfo` and `UserInfo` have multiple array variables which are important for the frontend, if this information is requested through a view function, a gas fee per byte will be theoretically charged making it possible for the transaction to run out of gas as there is a limit set by the RPCs.
Recommendation(s)	Consider adding functions which do not return these variables directly, paginate them or redesign the system.
Resolution	PARTIALLY RESOLVED Although the `PoolInfo` arrays cannot grow, the `DepositInfo` arrays can still grow to a point where they are too large for the RPC to return in one go.

Issue #13	Potential overflow in `withdraw` function
Severity	MEDIUM SEVERITY
Description	In the withdraw function, the safeTransfer function does `_denomination - _fee` which could in theory overflow. Due to the `anonymousTree` implementation presented to us this seems unlikely but this is still considered bad practice, especially as different implementations can currently be provided in the `addPool` function.
Recommendation(s)	Consider using SafeMath `.sub` instead of unchecked subtraction.
Resolution	ACKNOWLEDGED

Issue #14	Depositing funds without binding a note can result in lost funds
Severity	LOW SEVERITY
Description	If the user calls the `deposit` function without binding a note, these funds could be lost.
Recommendation(s)	Consider making this case impossible and double check that if either transaction fails on the frontend, this risk is clearly communicated to the user.
Resolution	ACKNOWLEDGED

Issue #15	Frontend can spy on secrets and when compromised could withdraw for any secret captured
Severity	LOW SEVERITY
Description	The withdrawal logic works similar to Tornado Cash where all that is required to withdraw is a secret (the note). This secret could be captured by the frontend and forwarded to a malicious party. This risk presents itself when the developer is untrusted or when the frontend is exploited.
Recommendation(s)	Consider using a simple frontend that is hosted on ipfs which can be easily audited by third parties. Consider taking steps to become a trusted party in the community like going through a KYC session with your team at a party the community trusts.
Resolution	ACKNOWLEDGED

Issue #16	`Initialize` might not update the `lastRewardBlock` correctly, causing pools to not start at the expected block
Severity	LOW SEVERITY
Description	In the `initialize` function, the new `startBlock` is set. However, the `startBlock` for the specific pools (`lastRewardBlock`) is only updated if it was set to zero beforehand. This is not always the case because it can be defined in `addPool`.
Recommendation(s)	Consider updating all `lastRewardBlock` variables.
Resolution	RESOLVED Since the ownership has been renounced and the pools have started, this issue is no longer relevant.

Issue #17	Masterchef will not work for LP tokens
Severity	LOW SEVERITY
Description	Because the `swapForTors` function has to be called, the masterchef cannot add LP tokens like traditional masterchefs can. This is because it does not make sense to add pairs containing other pairs.
Recommendation(s)	Consider checking if the path length is not zero before executing `swapForTors`
Resolution	RESOLVED Since the ownership has been renounced, this issue is no longer relevant.

Issue #18	When a pool is added, the `lastRewardBlock` can be set manually causing an exceptionally high mint if this is set to a block in the past
Severity	LOW SEVERITY
Description	When a pool is added, the `lastRewardBlock` can be set manually causing an exceptionally high mint if this is set to a block in the past
Recommendation(s)	Consider requiring this variable to be in the future or simply using the traditional masterchef function that sets it to the greater of the current block and the `startBlock`.
Resolution	RESOLVED Since the ownership has been renounced, this issue is no longer relevant.

Issue #19	No reentrancy guards on external functions
Severity	LOW SEVERITY
Description	It is considered good practice to include reentrancy guards once external functions grow in complexity like in this protocol.
Recommendation(s)	Consider adding reentrancy guards on external functions that have external calls. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol
Resolution	ACKNOWLEDGED

Issue #20	`getPoolInfo(uint256_pid)` can run out of gas
Severity	LOW SEVERITY
Description	The `getPoolInfo` function can return a significant number of bytes because of the arrays. If these arrays grow too large of a size the `getPoolInfo` call will run out of gas.
Recommendation(s)	Consider carefully if this state will ever be reached.
Resolution	RESOLVED Since the ownership has been renounced, these arrays are of fixed size.

Issue #21	`target.call.value` is deprecated
Severity	INFORMATIONAL
Description	Within the `functionCallWithValue` function of the address library, the more modern notation `target.call{value: value}(data)` is commented out.
Recommendation(s)	Consider replacing the deprecated notation with the notation that is currently considered best practice.
Resolution	ACKNOWLEDGED

Issue #22	The `INITIALIZED` variable is private
Severity	INFORMATIONAL
Description	Having private variables for important state makes auditing and reviewing more difficult. Especially within the farming space, being reviewable by third parties is important for investor confidence. Note that this is not as relevant in this case as most of the user functions will not work unless initialized.
Recommendation(s)	Consider setting the `INITIALIZED` variable as public.
Resolution	ACKNOWLEDGED

Issue #23	`addPool` casts the `_lpToken` to `ITorCoin` instead of `IBEP20` (typo)
Severity	INFORMATIONAL
Description	Within the `addPool` function, the lpToken is cast to `ITorCoin`. Although this is no issue, it may confuse reviewers.
Recommendation(s)	Consider casting this to `IBEP20`, which is the actual type of this variable.
Resolution	ACKNOWLEDGED

Issue #24	`findDenomination` may run out of gas
Severity	INFORMATIONAL
Description	The `findDemonimation` method uses a for loop which grows in size with the length of the denominations array. This may require transactions to revert due to reaching the gas limit.
Recommendation(s)	Consider using a `mapping(uint256 => uint256)` which maps amounts to their idx. Mappings their search operates in O(1) time.
Resolution	RESOLVED Since the ownership has been renounced, this array is of fixed size.

Issue #25	Public functions not used internally can be made external
Severity	INFORMATIONAL
Description	A significant part of the public functions are not used within the masterchef.
Recommendation(s)	Consider setting all public functions which are not used internally 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.
Resolution	ACKNOWLEDGED

2.5 Anonymous-tree Deploy

2.5.1 Privileged Roles

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

- `updateAnonymousTreeOwnership`
- `create`
- `bind`
- `unBind`
- `renounceOwnership`
- `transferOwnership`

2.5.2 Issues & Recommendations

Issue #26	Verifier not generated with secure randomization could result in loss of all funds
Severity	INFORMATIONAL
Description	`updateAnonymousTreeOwnership` can transfer an `_anonymousTree` ownership to another address, and can be called by the owner of the deploy contract. Owner can then bind/unbind.
Recommendation(s)	The owner of a `_anonymousTree` should always be the masterchef.
Resolution	PARTIALLY RESOLVED While the owner has been set to the Masterchef, the admin could have bound notes in before transferring ownership. The investor has to carefully go over the owner transactions to verify this.

2.6 TorCoin

2.6.1 Token Overview

The TorCoin is a token forked from the panther token. Its main custom feature is that it has a method that can be called once with 30 BNB to instantiate the two pairs. The token initially has a 4% transfer tax (max 10%) which is used completely to generate liquidity. Up to 100% of this transfer tax can be burned. The generated LP tokens from the transfer tax are also burned which means the developer can not dump these.

2.6.2 Privileged Roles

The following functions can be called by the operator of the token:

- `updateTransferTaxRate`
- `updateBurnRate`
- `updateMaxTransferAmountRate`
- `updateMinAmountToLiquify`
- `setExcludedFromAntiWhale`
- `updateSwapAndLiquifyEnabled`
- `updateTorSwapRouter`
- `transferOperator`

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

- `renounceOwnership`
- `transferOwnership`
- `mint`
- `initLiquify`

2.6.3 Issues & Recommendations

Issue #27	There is no minimum for `maxTransferAmount` (anti-whale).
Severity	HIGH SEVERITY
Description	This can be set to an unreasonably small amount to essentially make it extremely difficult, if not near impossible, to transfer tokens.
Recommendation(s)	Consider adding a reasonable minimum in, such as 0.5% of total supply. Non-zero is insufficient.
Resolution	RESOLVED The operator has been renounced.

Issue #28	`updateTorSwapRouter` could be used to siphon all liquidity fees or even turn the token into a honeypot
Severity	HIGH SEVERITY
Description	The operator can update the router that generates liquidity to an address or contract of choice. This 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(s)	Consider removing this function. If this is not possible, consider using an "operator" account which is behind a significantly longer timelock so investors can reasonably see this change coming and inspect the new router.
Resolution	RESOLVED The operator has been renounced.

Issue #29	All initial LP tokens are sent to the operator
Severity	LOW SEVERITY
Description	Investors are often uncomfortable with the developer owning all the LP tokens. In the `initLiquify` function, both pairs are simply sent to the operator, which is likely an EOA at the time of this function call.
Recommendation(s)	Consider calling this function before doing some form of presale if this is the plan. Then consider locking in the LP pairs in vesting contracts for investor confidence.
Resolution	RESOLVED The initial LP tokens have been burned: https://bscscan.com/tx/0xab4bc70b830e09e4941b48dc087961283faf8c3a0b4ee1c23d6b2a264238b402 https://bscscan.com/tx/0xdc6fa720381c319d5bb7d987f2a5a1e650cdf993d711e10bae0f585890ef7133

Issue #30	`_inInitLiquify` is private
Severity	LOW SEVERITY
Description	Having private variables for important state makes auditing and reviewing more difficult. Especially within the farming space, being reviewable by third parties is important for investor confidence.
Recommendation(s)	Consider setting the `_inInitLiquify` variable as public
Resolution	ACKNOWLEDGED

Issue #31	Token contract could build up dust BNB due to `addLiquidity`
Severity	LOW SEVERITY
Description	Due to the liquidity generating mechanism, some dust BNB which is not withdrawable may accumulate in the contract over time.
Recommendation(s)	Consider adding an `onlyOperator buybackAndBurn` function to use this dust to buy back and burn the native token. This aligns well with the project tokenomics.
Resolution	ACKNOWLEDGED

Issue #32	The developer is excluded from anti-whale
Severity	LOW SEVERITY
Description	During contract deployment, the account that deployed the contract is excluded from anti-whale.
Recommendation(s)	Consider including this account again to increase investor confidence in the project.
Resolution	RESOLVED Developer is included in antiwhale.

Issue #33	`initLiquify` could be abused if it is not called early on
Severity	INFORMATIONAL
Description	Due to the liquidity generating mechanism, some dust BNB which is not withdrawable may accumulate in the contract over time.
Recommendation(s)	Consider adding an `onlyOperator buybackAndBurn` function to use this dust to buy back and burn the native token. This aligns well with the project tokenomics.
Resolution	ACKNOWLEDGED

Issue #34	Exclude relevant contracts from anti-whale
Severity	INFORMATIONAL
Description	The masterchef should be excluded from anti-whale. Note that this was also done correctly on the previous iteration.
Recommendation(s)	Exclude the masterchef from anti-whale.
Resolution	ACKNOWLEDGED

Issue #35	`initLiquify` can be made `external`
Severity	INFORMATIONAL
Description	The `initLiquify` function 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 (https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices).
Recommendation(s)	Consider making this function external.
Resolution	ACKNOWLEDGED

Issue #37	`delegateBySig` can be frontrun and cause denial of service
Severity	INFORMATIONAL
Description	Currently if `delegateBySig` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up `delegateBySig` transactions in the mempool and execute them before a contract can. The issue with this is that the rest of said contract functionality would be lost as well. This could be a problem in case it would have been executed by a contract that would have rewarded you for your delegation for example.
Recommendation(s)	Consider adding the desired message sender in the `structhash` and requiring this desired sender to be equal to `msg.sender`. This reduces the problem to having the message sender be able to frontrun you which is okay if it is a reviewed contract.
Resolution	ACKNOWLEDGED

Issue #38	`target.call.value` is deprecated
Severity	INFORMATIONAL
Description	Within the `functionCallWithValue` function of the address library, the more modern notation `target.call{value: value}(data)` is commented out.
Recommendation(s)	Consider replacing the deprecated notation with the notation that is currently considered best practice.
Resolution	ACKNOWLEDGED

2.7 Timelock

The deployed address of the Timelock contract is 0xda1e55da00265ca9ef2ddbefd7c5c49ab31eafa9. The timelock is a standard clone of Compound Finance's timelock.

2.7.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY