PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For PolyArcadium

04 August 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 PolyArcadium, the second layer of the PolyWantsACracker farm. 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	PolyArcadium
URL	https://stadiumarcadium.farm
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

We have verified that the deployed contracts below match the final contracts that were provided to us in a zip archive.

Name	Contract	Live Code Match
MasterChef	0x9DD1fe32Aff4060c12E2b42961548876053187c6	MATCH
ArcadiumToken	0x3F374ed3C8e61A0d250f275609be2219005c021e	MATCH
MyFriendsToken	0xa509Da749745Ac07E9Ae47E7a092eAd2648B47f2	MATCH
ArcadiumReferral	0x2eA1955C58B195e6d43430c46a791B6934375E84	MATCH
L2LithSwap	0x5A4F563fe25Ca93bBE4Cbe8029499657b513F9C2	MATCH
PreMyFriends	0xe7280E6b0E8589c70576d3bd22BBe245d481Cf99	MATCH
PreArcadium	0x273FE932bC9792b6cDA3C837A5231CE3F0Da21d9	MATCH
L2TokenRedeem	0x3AB5a9AF4eEaa173D83a2AE36c81FF84cB6FEd06	MATCH
AddLiquidityHelper	0x75eF87C8994c6c7Bc07C80fEf255AFb32dCE77C2	MATCH
RHCPToolBox	0xadd074dd2a2ce3552768fa75152ad53ac2842edc	MATCH
Timelock	0x6a8af1dbFdb32dAc39BF8A386c03cC8857a107a8	MATCH
Locker	0x3437008342c7332C82c30558ab45400C6362c7F4	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	9	9	-	-
Medium	7	7	-	-
Low	17	17	-	-
Informational	26	26	-	-
Total	**59**	**59**	**0**	**0**

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 MasterChef

ID	Severity	Summary	Status
01	MEDIUM	The `nonDuplicated` mechanism is not properly implemented	RESOLVED
02	MEDIUM	`pendingMyFriends` currently miscalculates the pending reward amount by not excluding the MyFriends token pool	RESOLVED
03	LOW	Undetermined state while myFriendsPID is not yet set	RESOLVED
04	LOW	Skim limit might be excessive for expensive tokens	RESOLVED
05	LOW	Pool `tokenType` should not be changeable	RESOLVED
06	LOW	`autoUpdateArcadiumGradient` is not called on withdrawal	RESOLVED
07	LOW	There are no sanity checks on the `setReferralAddress`	RESOLVED
08	LOW	`myFriends`, `arcadium` and `arcadiumToolbox` are private	RESOLVED
09	INFORMATIONAL	`pendingArcadium` and `pendingMyFriends` will revert in case totalAllocPoint is zero	RESOLVED
10	INFORMATIONAL	`add` and `set` can be made external	RESOLVED
11	INFORMATIONAL	Gas efficiency: Unnecessary `if` statement in `deposit`	RESOLVED
12	INFORMATIONAL	Lack of event for `setArcadiumReferral`	RESOLVED
13	INFORMATIONAL	Typos in documentation	RESOLVED
14	INFORMATIONAL	`usdcRewardCurrency`, `founderFinalLockupEndBlock`, `myFriends`, `arcadium`, `arcadiumToolbox`, `startBlock`, `gradient2endBlock` and `gradient3endBlock` can be made `immutable`	RESOLVED
15	INFORMATIONAL	`initialFounderMyFriendsStake`, `gradient2EndEmmissions` and `gradient3EndEmmissions` can be made constant	RESOLVED
16	INFORMATIONAL	Constructor gradient calculations do not reuse the toolbox functionality to calculate the gradient	RESOLVED

1.3.2 ArcadiumToken

ID	Severity	Summary	Status
17	HIGH	`updateArcadiumSwapRouter` could be used to steal the deposit fees by updating to a malicious router	RESOLVED
18	MEDIUM	Generated liquidity tokens are sent to the dev address	RESOLVED
19	MEDIUM	Token contract is not automatically excluded from transfer tax	RESOLVED
20	MEDIUM	Miscalculation in the `arcadiumToLPwith` integer in the liquidity generation function	RESOLVED
21	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	RESOLVED
22	LOW	Tokens without a relevant pair might revert the transaction	RESOLVED
23	LOW	`addLiquidityHelper` and `arcadiumToolbox` are private	RESOLVED
24	INFORMATIONAL	`swapDepositFeeForTokensInternal` has a misleading return statement	RESOLVED
25	INFORMATIONAL	`usdcRewardCurrency` and `myFriends` can be made immutable	RESOLVED
26	INFORMATIONAL	Lack of event for `transferOperator`, `updateArcadiumSwapRouter`, `updateFeeMaps` and `updateSwapAndLiquifyEnabled`	RESOLVED

1.3.3 MyFriendsToken

ID	Severity	Summary	Status
27	HIGH	`updateArcadiumSwapRouter` could be used to steal the deposit fees by updating to a malicious router	RESOLVED
28	HIGH	`checkTokenStockpileUSDCValue` makes an unnecessary decimal adjustment	RESOLVED
29	MEDIUM	MyFriendsToken does not use the RHCPToolbox contract to calculate the USDC value	RESOLVED
30	INFORMATIONAL	Gas efficiency: Redundant `balanceOf` check in distribute function	RESOLVED
31	INFORMATIONAL	`usdcSwapThreshold` can be made `constant`	RESOLVED
32	INFORMATIONAL	`usdcRewardCurrency` can be made `immutable`	RESOLVED
33	INFORMATIONAL	Lack of events for `distribute`, `convertDepositFeesToUSDC`, and `transferUSDCToUser`	RESOLVED

1.3.4 ArcadiumReferral

ID	Severity	Summary	Status
34	LOW	The owner of the Referral contract can overwrite themselves as the referrer for all users using the `recordReferral` function	RESOLVED

1.3.5 PreMyFriends

ID	Severity	Summary	Status
35	LOW	The presale will take slightly longer since the average block interval now exceeds two seconds on the Polygon network	RESOLVED
36	INFORMATIONAL	`preArcadiumAddress` can be made `immutable`	RESOLVED
37	INFORMATIONAL	Errors sometimes indicate the wrong token due to copy-pasting them	RESOLVED

1.3.6 PreArcadium

No issues found.

1.3.7 L2LithSwap

ID	Severity	Summary	Status
38	HIGH	The `preMyFriendsRemaining` and `preArcadiumRemaining` values are not adjusted after swaps	RESOLVED
39	MEDIUM	The `feeAddress` could participate in the swapping and claim the whole allocation	RESOLVED
40	LOW	The swap period will take slightly longer since the average block interval now exceeds two seconds on polygon	RESOLVED
41	INFORMATIONAL	Comment says that the `feeAddress` is the burn address	RESOLVED
42	INFORMATIONAL	`preArcadiumAdddress` and `preMyFriendsAddress` can be made `immutable`	RESOLVED
43	INFORMATIONAL	Errors sometimes indicate the wrong token due to copy-pasting them	RESOLVED

1.3.8 L2TokenRedeem

ID	Severity	Summary	Status
44	HIGH	`sendUnclaimedToFeeAddress` sends presale MyFriends tokens instead of the actual MyFriends tokens to the fee address	RESOLVED
45	LOW	`L2LithSwap`, `preMyFriends`, `preArcadiumAddress`, `myFriendsAddress` and `arcadiumAddress` can be made `immutable`	RESOLVED

1.3.9 AddLiquidityHelper

ID	Severity	Summary	Status
46	HIGH	`arcadiumAddress` can be changed after initialisation which can lead to siphoning of tokens sent to the contract in case the dev abuses this	RESOLVED
47	LOW	Excess tokens will get stuck in the `AddLiquidityHelper` during liquidity addition	RESOLVED
48	INFORMATIONAL	Router is not modifiable while the documentation indicates that it is	RESOLVED
49	INFORMATIONAL	Functionality is inefficient with transfer-tax tokens	RESOLVED
50	INFORMATIONAL	The `AddLiquidityHelper` can be used by anyone	RESOLVED
51	INFORMATIONAL	`arcadiumAddress` and `addArcadiumLiquidity` can be made `immutable`	RESOLVED
52	INFORMATIONAL	Lack of events for `setArcadiumAddress`	RESOLVED

1.3.10 RHCPToolbox

ID	Severity	Summary	Status
53	HIGH	`convertToTargetValueFromPair` makes an unnecessary decimal adjustment	RESOLVED
54	HIGH	Transactions will revert if the end emission ever reaches zero	RESOLVED
55	HIGH	Wrong emission calculation results in about half of the expected emission rate	RESOLVED
56	LOW	Operator is `private`	RESOLVED
57	LOW	Operator is `immutable`	RESOLVED
58	LOW	Router is changeable which could cause the contract to break in case it is changed to a malicious contract	RESOLVED
59	INFORMATIONAL	`startBlock` can be made `immutable`	RESOLVED

1.3.11 Timelock

No issues found.

1.3.12 Locker

No issues found.

2 Findings

2.1 MasterChef

Deposit fees in the Masterchef are limited to 4.01%. There is a fixed 6% referral commission, where 3% goes to the referrer and 3% goes to the referral.

There are two founder addresses hardcoded in the Masterchef. When these addresses harvest, 50% of their Arcadium rewards and 100% of their MyFriends rewards are burned. There's also a special vesting mechanism that only allows these two accounts to withdraw funds as time passes.

In case the Masterchef has more tokens than are deposited (eg. through accidental transfers or reflection), the surplus is divided in two and each half is transferred to one of the native tokens. This mechanism does not apply to the native pool.

Stakers in the MyFriends pool receive the USDC dividends from deposit fees. 100% of liquidity pool (LP) tokens received from deposit fees are sent to the ArcadiumToken contract and broken up there. 25% of regular tokens received from the deposit fees are sent to the ArcadiumToken contract, and 75% of deposit fees received are sent to the MyFriendsToken contract. The MyFriendsToken contract converts the deposit fees received into USDC and provides these to MyFriends stakers as dividends. About 82% of the burned Arcadium is distributed back to MyFriends holders.

MyFriends is emitted at a constant rate of 0.032 MyFriends per block to all other pools until the supply in the MyFriends contract has been exhausted.

2.1.2 Privileged Roles

The ownership of the token contract has been transferred to the Masterchef. The following functions can be called by the owner of the Masterchef:

- `renounceOwnership`
- `transferOwnership`
- `add`
- `set`
- `setArcadiumReferral`

2.1.3 Issues & Recommendations

Issue #01	**The `nonDuplicated` mechanism is not properly implemented**
Severity	MEDIUM SEVERITY
Description	The Masterchef contains a mechanism that disallows adding a pool twice using the `nonDuplicated` modifier. However, when a pool is added, it is never set as "existing" in the `poolExistence` mapping, thus the `nonDuplicated` modifier will always say that a pool is not a duplicated pool. The implications of this omission seem rather minimal since the `nonDuplicated` check is just a courtesy check and adding a pool with the same staking token twice would not really change the business logic of the contract. However, since this is an explicit function in the contract and it is not working at all, we decided medium severity is appropriate for this issue.
Recommendation	Consider adding `poolExistence[_lpToken] = true;` to the `add` function.
Resolution	RESOLVED The recommended code has been added to the `add` function.

Issue #02	**`pendingMyFriends` currently miscalculates the pending reward amount by not excluding the MyFriends token pool**
Severity	MEDIUM SEVERITY
Description	In the reward payout function, the MyFriends pool is excluded in the calculation of the `totalAllocPoints` since this pool does not receive MyFriends rewards. However, in the pending function this exclusion is not made.
Recommendation	Consider adding this exclusion to the pending function by replacing `totalAllocPoint` with `(totalAllocPoint - poolInfo[myFriendsPID].allocPoint)`.
Resolution	RESOLVED

Issue #03	**Undetermined state while `myFriendsPID` is not yet set**
Severity	LOW SEVERITY
Description	While the MyFriends pool is not yet added, the `myFriendsPID` variable points to pool zero. This could result in strange behavior on the USDC payout mechanism.
Recommendation	Consider setting the MyFriends pool in the constructor to avoid this intermediary state.
Resolution	RESOLVED The following line has been added to instantiate the pool immediately at deployment: `add(0, 6000, _myFriends, 0, false);`

Issue #04	Skim limit might be excessive for expensive tokens
Severity	LOW SEVERITY
Description	The skim mechanism where tokens that were received through reflection or accidental transfers are transferred out is triggered only if there are more than 100 tokens in the Masterchef. This might be a very high amount for tokens like BTC.
Recommendation	Consider whether this will be an issue and if so, consider adding a skim limit in the `poolInfo`.
Resolution	RESOLVED The skim limit is now negligibly low, so even the more expensive currencies are being skimmed often.

Issue #05	Pool `tokenType` should not be changeable
Severity	LOW SEVERITY
Description	The `tokenType` of a pool indicates whether it is an LP (`tokenType` 1) or a normal token (`tokenType` 0). We are unsure in what instances the `tokenType` should be changed and thus recommend removing the option from the `set` function. The one potential case we do see is when the router changes, then an argument can be made that the original LPs are now no longer real LPs for the system.
Recommendation	Consider whether the adjustment of token types is really necessary and if not, consider removing the option to change them to reduce the governance privileges further.
Resolution	RESOLVED This parameter is removed completely.

Issue #06	`autoUpdateArcadiumGradient` is not called on withdrawal
Severity	LOW SEVERITY
Description	The `autoUpdateArcadiumGradient` function is used to adjust the emissions rate. However, it is only called on deposits while emissions also happen on withdrawals. This might cause the emission schedule to slightly differ from the expected one when people do not deposit but only withdraw in the adjustment period.
Recommendation	Consider also doing the `autoUpdateArcadiumGradient` call in the withdrawal function.
Resolution	RESOLVED The developer has explained how they carefully decided in favor of this behavior as to always allow withdrawals to proceed.

Issue #07	There are no sanity checks on the `setReferralAddress`
Severity	LOW SEVERITY
Description	A lot of functionality can break if the referral address is updated to a value that is not a referral contract. Furthermore, the referral address could be upgraded to a malicious contract that makes the dev the referral of every user, effectively minting 3% of the emissions to the dev.
Recommendation	Consider making the referral address non upgradeable (only settable once) to ensure that functionality can never break. We rarely ever see a project updating their referral after it is initially set.
Resolution	RESOLVED The `arcadiumReferral` can only be set once.

Issue #08	`myFriends`, `arcadium` and `arcadiumToolbox` are `private`
Severity	LOW SEVERITY
Description	Not marking important variables like contracts as `public` makes it difficult for third-party reviewers to verify that these variables are set correctly. In case they are set incorrectly (for example to a malicious contract) this could lead to very bad results for the investors.
Recommendation	Consider making the `myFriends`, `arcadium` and `arcadiumToolbox` variables public so anyone can verify that these are set correctly.
Resolution	RESOLVED

Issue #09	`pendingArcadium` and `pendingMyFriends` will revert in case `totalAllocPoint` is zero
Severity	INFORMATIONAL
Description	The GUI functions to fetch the pending reward balances will revert in case the `totalAllocPoint` variable is set to zero. These functions are only used for display purposes but it is still valuable to fix this issue since the frontend could crash in this situation.
Recommendation	Consider adding a `totalAllocPoint` check to the already present if statement: `if (block.number > pool.lastRewardBlock && lpSupply != 0 && totalAllocPoint != 0) {`
Resolution	RESOLVED The recommended code update has been implemented in both pending functions.

Issue #10	`add` and `set` can be made external
Severity	INFORMATIONAL
Description	The `add` and `set` 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 (https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices).
Recommendation	Consider making these functions `external`.
Resolution	RESOLVED `set` has been made `external` and since `add` is now used in the constructor, it must remain `public`.

Issue #11	Gas efficiency: Unnecessary `if` statement in `deposit`
Severity	INFORMATIONAL
Description	Within the `deposit` function, first a check is done if the `tokenType` is 1 (meaning it's an LP token). Then a check is done if the `tokenType` is 0 (a normal token). This other check will always happen regardless of whether the first one passed, wasting unnecessary gas.
Recommendation	Consider using `else if` in the second statement.
Resolution	RESOLVED The statement has been adjusted to use `else if`.

Issue #12	Lack of event for `setArcadiumReferral`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for `setArcadiumReferral`.
Resolution	RESOLVED

Issue #13	Typos in documentation
Severity	INFORMATIONAL

Description	Within the contract, among others, the following erroneous comments can be found:
	`// and deployed are on a cheap gas network POLYGON (MATIC)`
	`// Return if address a founder address`
	`// we only allow 80 pools to be upda`
	`//n Cannot skim any tokens we use for staking rewards.`
	`// Return how much myFriends should be saked by the founders at any time.`
	`// No MyFriends withdrawals before farmiing`

Recommendation	Consider fixing these typos.
Resolution	RESOLVED

Issue #14	**`usdcRewardCurrency`, `founderFinalLockupEndBlock`, `myFriends`, `arcadium`, `arcadiumToolbox`, `startBlock`, `gradient2endBlock` and `gradient3endBlock` can be made immutable**
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the immutable keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `usdcRewardCurrency`, `founderFinalLockupEndBlock`, `myFriends`, `arcadium`, `arcadiumToolbox`, `startBlock`, `gradient2endBlock` and `gradient3endBlock` explicitly immutable.
Resolution	RESOLVED

Issue #15	`initialFounderMyFriendsStake`, `gradient2EndEmmissions` and `gradient3EndEmmissions` can be made constant
Severity	INFORMATIONAL
Description	Variables that never change throughout the lifecycle of the contract can be marked with the `constant` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `initialFounderMyFriendsStake`, `gradient2EndEmmissions` and `gradient3EndEmmissions` explicitly `constant`.
Resolution	RESOLVED

Issue #16	Constructor gradient calculations do not reuse the toolbox functionality to calculate the gradient
Severity	INFORMATIONAL
Description	Within the RHCPToolBox, the `calcEmissionGradient` helper function is responsible for calculating the gradient of the emission schedule (how much the emission rate should increase each block). However, within the Masterchef constructor, this code is not reused and instead present in a slightly modified form. In case this business logic is ever updated, this might cause divergence between the two implementations. This follows from the well-known programming principle called don't repeat yourself.
Recommendation	Consider reusing the toolbox for calculating the gradient in the constructor.
Resolution	RESOLVED The RHCPToolBox is now used, reducing the code complexity greatly.

2.2 ArcadiumToken

The ArcadiumToken is a fork of the Panther token with custom functionality. On deployment, 325,000 tokens are pre-minted to an EOA.

The transfer tax is initially set to 6.66% and can be increased to 8.88% on transfers to or from specific contracts. This feature can be used to tax selling transactions more heavily than buying transactions (which seems to be the use case). The tax percentages can be adjusted but both values cannot exceed 10.01%. A fixed 54.95% of the transfer tax is burned while the remaining amount is used to generate liquidity.

The liquidity generation mechanism is interesting and unique in that part of the deposit fees sent to Arcadium (25%) is sold off for Matic, and this Matic is then paired with the transfer-tax fraction for LP generation. If the values are not in balance, Arcadium will either be bought or sold for Matic to achieve a balance for LP generation. This mechanism results in either providing buying or selling pressure, depending on the amount of deposit fees compared to Arcadium transfers.

The owner of the token is the Masterchef and the operator of the token is the Timelock.

2.1.1 Token Overview

Address	0xeB0d362862982110B6e146d9bD7515800a12bCc0
Token Supply	1,000,000 (one million) + referral mints
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	6.66% to 8.66%

2.2.1 Privileged Roles

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

- `renounceOwnership`
- `transferOwnership`
- `mint`
- `swapDepositFeeForETH`
- `updateSwapAndLiquifyEnabled`
- `updateTransferTaxRate`
- `updateExcludeMap`
- `updateExtraMap`
- `updateArcadiumSwapRouter`
- `transferOperator`

2.2.2 Issues & Recommendations

Issue #17	`updateArcadiumSwapRouter` could be used to steal the deposit fees by updating to a malicious router
Severity	HIGH SEVERITY
Description	The owner 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 and thus siphons all deposit fees.
Recommendation	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 arcadium router can now only be set once.

Issue #18	Generated liquidity tokens are sent to the `dev` address
Severity	MEDIUM SEVERITY
Description	The generated liquidity tokens are sent to the dev address – this might scare off investors since it creates dumping risk.
Recommendation	Consider carefully what the appropriate destination of this generated liquidity will be – in case it is not the developer's wallet, it might boost investor confidence if this destination is implemented directly in the code. Either way, it might still be valuable not to hard-code this address in case the client decides to change this address to, for example, a timelock at a later date.
Resolution	RESOLVED The added liquidity are now immediately burned within the code.

Issue #19	Token contract is not automatically excluded from transfer tax
Severity	MEDIUM SEVERITY
Description	The token is not excluded automatically from the transfer tax. If this is forgotten, each swap interaction might create taxable events. This issue is marked as medium risk as the `addLiquidityHelper` interactions will seriously malfunction (revert) if they are taxed.
Recommendation	In case the token liquidity generating actions should be excluded from the transfer tax, consider adding the token address to the exclusion map within the constructor itself. This saves a deployment step.
Resolution	RESOLVED The `addLiquidityHelper` is explicitly excluded from any transfer tax.

Issue #20	**Miscalculation in the `arcadiumToLPwith` integer in the liquidity generation function**
Severity	MEDIUM SEVERITY
Description	At the end of `swapAndLiquify`, the `arcadiumToLPwith` contains a neligible miscalculation: `(unmatchedArcadium - unmatchedArcadiumToSwap)` should simply be `unmatchedArcadiumToSwap`. This should not cause significant issues since it can be at most 1 unit off and in this case it would always allow 1 less unit to be added to liquidity, which should not be an issue. However, we still think it is worthwhile to mark this issue as medium risk as this code is needlessly complex. It should probably be simplified to simply add the current balances in the contract as liquidity.
Recommendation	At the end of the whole liquidity generation sequence in `swapAndLiquify`, consider simply querying `address(this).balance` and `ERC20(address(this)).balanceOf(address(this))` to figure out the amounts that should be used for liquidity generation. In case there would be an imbalance, we have recommended that the `addLiquidityHelper` should refund it in another issue.
Resolution	RESOLVED The business logic has been isolated to the AddLiquidityHelper and contains the recommended resolution..

Issue #21	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef
Severity	LOW SEVERITY
Description	The mint function could be used to pre-mint tokens for legitimate uses including, but not limited to, the injection of initial liquidity, token presale, or airdrops (this practice is prevalent amongst less-reputable projects, unfortunately). As token ownership has already been transferred to the Masterchef, the main risk that remains is whether any tokens were pre-minted to the project owner prior to transferring ownership.
Recommendation	Consider being forthright if this mint function has been used by letting your community know how much was minted, where they are currently stored, if a vesting contract was used for token unlocking, and finally the purpose of the mints.
Resolution	RESOLVED The developers have gone out of their way to resolve this issue by making ownership transferable exactly once, minting is only allowed once it is transferred. This means that at most they could transfer ownership to a malicious contract, but as soon as it is set to the Masterchef, investors can be certain that no malicious minting has ever been done.

Issue #22	Tokens without a relevant pair might revert the transaction
Severity	LOW SEVERITY
Description	In case a token is swapped but no relevant pair (eg. it is not yet paired with matic) can be found to swap it for, the swap transactions will revert.
Recommendation	Consider ensuring that each token added is paired with Matic on the AMM or to wrap the swap logic in a try-catch clause (https://blog.ethereum.org/2020/01/29/solidity-0.6-try-catch/).
Resolution	RESOLVED The recommended try-catch logic has been implemented.

Issue #23	`addLiquidityHelper` and `arcadiumToolbox` are private
Severity	LOW SEVERITY
Description	Not marking important variables like contracts as `public` makes it difficult for third-party reviewers to verify that these variables are set correctly. In case they are set incorrectly (for example to a malicious contract), this could lead to very bad results for the investors.
Recommendation	Consider making the `addLiquidityHelper` and `arcadiumToolbox` variables `public` so anyone can verify that these are set correctly.
Resolution	RESOLVED

Issue #24	`swapDepositFeeForTokensInternal` has a misleading `return` statement
Severity	INFORMATIONAL
Description	Within the `swapDepositFeeForTokensInternal` function, the following `return` statement can be found: `return swapLpTokensForFee(token, totalTokenBalance);` However, this `return` statement is considered misleading since the functions do not return a value. This can confuse third-party reviewers.
Recommendation	Although we are sure writing this code in this way saves a few lines of code, consider writing this clause in a multi-line fashion so as to not confuse third-party reviewers into thinking that these functions might have return variables.
Resolution	RESOLVED The `return` statement has been moved to a separate line.

Issue #25	`usdcRewardCurrency` and `myFriends` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `usdcRewardCurrency` and `myFriends` explicitly immutable.
Resolution	RESOLVED The variables along with `addLiquidityHelper` and `arcadiumToolBox` are now immutable.

Issue #26	Lack of event for `transferOperator`, `updateArcadiumSwapRouter`, `updateFeeMaps` and `updateSwapAndLiquifyEnabled`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for `transferOperator`, `updateArcadiumSwapRouter`, `updateFeeMaps` and `updateSwapAndLiquifyEnabled`.
Resolution	RESOLVED

2.3 MyFriendsToken

The MyFriendsToken is one of the two primary Layer 2 tokens. During deployment, 80,000 tokens are minted to the dev wallet and a remaining 20,000 tokens are minted to the token address itself. These remaining tokens can be distributed by the owner which is initially the deployer but should later be the Masterchef.

The owner of the token (which is currently the Masterchef) can also transfer USDC out of the contract (apart from the token itself). The operator of the token is behind a 6 hour Timelock.

2.3.1 Token Overview

Address	0xd871C50eF8865F0f83ab2b6E595fE2C3A091D5e0
Token Supply	100,000 (one hundred thousand)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.3.2 Privileged Roles

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

- `renounceOwnership`
- `transferOwnership`
- `distribute`
- `convertDepositFeesToUSDC`
- `transferUSDCToUser`
- `updateArcadiumSwapRouter`
- `transferOperator`

2.3.2 Issues & Recommendations

Issue #27	`updateArcadiumSwapRouter` could be used to steal the deposit fees by updating to a malicious router
Severity	HIGH SEVERITY
Description	The owner 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 and thus siphons all deposit fees.
Recommendation	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 router can now be set only once.

Issue #28	`checkTokenStockpileUSDCValue` makes an unnecessary decimal adjustment
Severity	HIGH SEVERITY
Description	In the `checkTokenStockpileUSDCValue` function, a decimal adjustment is made on the resulting number. However, since the reserves already account for the actual decimals these tokens have, no decimal adjustment should be made. Note that there is also a multiplication where a division should occur. But since this logic will have to go we will not turn this in a separate issue.
Recommendation	Simply using the `RHCPToolbox` to calculate the USDC value is the recommended route to fixing this issue.
Resolution	RESOLVED This function has been removed in favor of reusing the RHCPToolbox utility contract.

Issue #29	**MyFriendsToken does not use the `RHCPToolbox` contract to calculate the USDC value**
Severity	MEDIUM SEVERITY
Description	The developer has developed an intricate system to calculate the USDC value of any token, and this functionality is stored within the `RHCPToolbox` contract. However, for some reason within this contract, the pricing functionality is reimplemented without relying on the `RHCPToolbox`.
Recommendation	Consider using the `RHCPToolbox` to calculate the USDC value.
Resolution	RESOLVED The RHCPToolbox is now reused in this contract, greatly simplifying the business logic.

Issue #30	**Gas efficiency: Redundant `balanceOf` check in `distribute` function**
Severity	INFORMATIONAL
Description	The `distribute` function contains a check that returns zero in case the balance within the contract is zero. This check is redundant since the code will end up in the exact same state even without this check.
Recommendation	Consider removing this check to make this function slightly shorter and more gas efficient.
Resolution	RESOLVED

Issue #31	`usdcSwapThreshold` can be made `constant`
Severity	INFORMATIONAL
Description	Variables that never change throughout the lifecycle of the contract can be marked with the `constant` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `usdcSwapThreshold` explicitly `constant`.
Resolution	RESOLVED

Issue #32	`usdcRewardCurrency` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `usdcRewardCurrency` explicitly `immutable`.
Resolution	RESOLVED

Issue #33	**Lack of events for `distribute`, `convertDepositFeesToUSDC`, and `transferUSDCToUser`**
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for `setGovburnRateAddress`, `setBonusAddress`, `setburnRate`, `setbonusRate` and `addWhiteLstAddress`.
Resolution	RESOLVED

2.4 ArcadiumReferral

The operator of the ArcadiumReferral contract is the Masterchef.

2.4.1 Privileged Roles

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

- `renounceOwnership`
- `transferOwnership`
- `updateOperator`
- `recordReferral`

2.4.2 Issues & Recommendations

Issue #34	**The owner of the Referral contract can overwrite themselves as the referrer for all users using the `recordReferral` function**
Severity	LOW SEVERITY
Description	The operator should only be the Masterchef contract. However, the owner of the Referral contract has privileges that may be abused. The following steps detail how the owner can make themselves the referrer for all users : 1. Owner of the Referral contract calls `updateOperator` to add themselves as an operator. 2. As an operator, they then call `recordReferral` with their own wallet address as the referrer for each user. 3. This results in their wallet address being minted potentially large amounts of referral commission.
Recommendation	There are 3 possible recommendations : 1. Consider making the `updateOperator` function callable only once. 2. Setting only the Masterchef as an operator in the modifier, and removing the `updateOperator` function. 3. Calling the `updateOperator` to set the Masterchef as operator, then renouncing ownership of the Referral contract such that it cannot be called in the future.
Resolution	RESOLVED The client has adjusted the functionality with safety in mind: There can now only be one operator and the address can only be set once. There is thus no way to mess with these values.

2.5 PreMyFriends

The PreMyFriends contract is the presale contract for both the MyFriends and Arcadium token. It accepts USDC and will proportionally give MyFriends and Arcadium tokens in return. Each wallet can participate with up to $5,000 USDC.

The owner of the contract can adjust the presale prices until 4 hours before the presale starts within the following limited ranges:

- Between 0.0007 and 0.02 MyFriends/USDC
- Between 0.004 and 0.11 Arcadium/USDC

The presale limit is $750,000 which cannot be changed. The total number of tokens available for both presales will be 250,000 Arcadium and 45,000 MyFriends.

The presale is scheduled to take 2 days. The contract mints an initial 80,000 presale MyFriends tokens to the `feeAddress` and a balance of both presale MyFriends and presale Arcadium tokens are required for the contract to function. All presale funds are sent to the `feeAddress` which is an externally-owned account (EOA).

2.5.1 Privileges

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

- `renounceOwnership`
- `transferOwnership`
- `setStartBlock`
- `setSaleINVPriceE35`

2.5.2 Issues & Recommendations

Issue #35	**The presale will take slightly longer since the average block interval now exceeds two seconds on the Polygon network**
Severity	LOW SEVERITY
Description	The average block interval has increased to about 2.4 seconds recently and is quite volatile in recent days. The presale could thus potentially take 10 hours longer than expected since the timing is based on blocks.
Recommendation	There are a few options: 1. It is not a problem that the presale might take 10 hours longer. 2. Adjust the `oneHourMatic` variable with the expected blocks per hour (about 1500 blocks but could be even less considering the recent trend). 3. Using the block timestamp instead of the block number to account for the duration.
Resolution	RESOLVED The client has adjusted the block time to the recommended 1500 blocks per hour. However, due to network volatility it is of course impossible to exactly time this to be two hours, except by using the block timestamp.

Issue #36	`preArcadiumAddress` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `preArcadiumAdddress` explicitly `immutable`.
Resolution	RESOLVED

Issue #37	Errors sometimes indicate the wrong token due to copy-pasting them
Severity	INFORMATIONAL

Description	Due to accidental copy-paste, certain `revert` statements refer to the wrong token:
	`require(IERC20(preArcadiumAddress).balanceOf(address(this)) > 0, "No more PreMyFriends left! Come back next time!");`
	`require(IERC20(preArcadiumAddress).transfer(msg.sender, preArcadiumPurchaseAmount), "failed sending preMyFriends");`
	These errors should of course be adjusted to indicate that they affect the `PreArcadium` token.

Recommendation	Consider adjusting the error string to indicate the correct token.
Resolution	RESOLVED The comments have been updated.

2.6 PreArcadium

The PreArcadium contract is a very simple ERC20 token without governance. During deployment, it pre-mints 250,000 tokens to 0x3a1D1114269d7a786C154FE5278bF5b1e3e20d31 (the dev wallet).

2.6.1 Issues & Recommendations

No issues found.

2.7 L2LithSwap

The L2LithSwap contract trades L1 Lithium tokens for both the MyFriends and Arcadium tokens. It accepts L1 Lithium tokens and will proportionally give MyFriends and Arcadium tokens in return. The total limit is 100,000 Lithium and there is no per-wallet limit. These Lithium tokens are however not burned; instead they are sent to the fee address.

The swap is scheduled to take 2 days. The contract requires a balance of both presale MyFriends and presale Arcadium tokens to function.

2.7.1 Privileges

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

- `renounceOwnership`
- `transferOwnership`
- `setSaleINVPriceE35`
- `setStartBlock`

2.7.2 Issues & Recommendations

Issue #38	**The `preMyFriendsRemaining` and `preArcadiumRemaining` values are not adjusted after swaps**
Severity	HIGH SEVERITY
Description	The `swapLithForPresaleTokensL2` function is used to swap Lithium into the two L2 tokens. The contract nicely keeps track of the remaining balances to give away through two variables, `preMyFriendsRemaining` and `preArcadiumRemaining`. However, during the swap transaction, in contrast to the presale contract, these two balances are not changed. This will result in a miscalculation in the L2TokenRedeem `sendUnclaimedToFeeAddress` function.
Recommendation	Consider deducting these balances properly like in the PreMyFriends contract.
Resolution	RESOLVED The client has added the balance adjustment logic in the swap function.

Issue #39	**The `feeAddress` could participate in the swapping and claim the whole allocation**
Severity	MEDIUM SEVERITY
Description	The Lithium raised in the presale is sent to the `feeAddress` immediately. This means that if the `feeAddress` were to call the swap function, they could keep calling it over and over again since they always receive back the sent Lithium. This might scare investors that do not trust the fee address holder as much.
Recommendation	Consider only allowing the `feeAddress` to withdraw the raised Lithium after the presale has ended. This way, there is no way for the `feeAddress` to execute this exploit.
Resolution	RESOLVED The contract now locks in the raised Lithium until the presale ends. After the presale has ended, the `feeAddress` can withdraw this Lithium.

Issue #40	**The swap period will take slightly longer since the average block interval now exceeds two seconds on polygon**
Severity	LOW SEVERITY
Description	The average block interval has increased to about 2.4 seconds recently and is quite volatile these days. The swap period could thus potentially take 10 hours longer than expected since the timing is based on blocks.
Recommendation	There are a few options: 1. It is not a problem that the presale might take 10 hours longer. 2. Adjust the `oneHourMatic` variable with the expected blocks per hour (about 1500 blocks but could be even less considering the recent trend). 3. Using the block timestamp instead of the block number to account for the duration.
Resolution	RESOLVED The client has adjusted the block time to the recommended 1500 blocks per hour. However, due to network volatility, it is of course impossible to exactly time this to be two hours, except by using the block timestamp.

Issue #41	Comment says that the `feeAddress` is the burn address
Severity	INFORMATIONAL
Description	Above the fee address declaration, the following comment can be found: `// Burn address` However, this address is of course not a burn address and this comment is thus obsolete.
Recommendation	Consider removing this deprecated comment.
Resolution	RESOLVED

Issue #42	`preArcadiumAdddress` and `preMyFriendsAddress` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `preArcadiumAdddress` and `preMyFriendsAddress` explicitly `immutable`.
Resolution	RESOLVED

Issue #43	Errors sometimes indicate the wrong token due to copy-pasting them
Severity	INFORMATIONAL

Description	Due to accidental copy paste, certain revert statements indicate the wrong token:
	`require(IERC20(preArcadiumAddress).balanceOf(address(this)) > 0, "No more PreMyFriends left! Come back next time!");`
	`require(IERC20(preArcadiumAddress).transfer(msg.sender, preArcadiumPurchaseAmount), "failed sending preMyFriends");`
	These errors should of course be adjusted to indicate that they affect the PreArcadium token.

Recommendation	Consider adjusting the error string to indicate the correct token.
Resolution	RESOLVED

2.8 L2TokenRedeem

The L2TokenRedeem contract allows users to swap out the MyFriends and Arcadium presale tokens for the actual tokens. The presale tokens are then burned. After the presale ends, the unclaimed L2 tokens can be sent to the admin.

2.8.1 Privileges

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

- `renounceOwnership`
- `transferOwnership`
- `setStartBlock`
- `sendUnclaimedsToFeeAddress`

2.8.2 Issues & Recommendations

Issue #44	`sendUnclaimedToFeeAddress` sends presale MyFriends tokens instead of the actual MyFriends tokens to the fee address
Severity	HIGH SEVERITY
Description	The intention of the `sendUnclaimedToFeeAddress` function is to withdraw the unclaimed final L2 tokens to the fee address. However, currently the code contains the following line to send the MyFriends tokens out: `preMyFriends.transfer(feeAddress, wastedPreMyFriendsTokend);` We assume this is supposed to be the L2 token and not the presale token.
Recommendation	Consider changing this transfer to `IERC20(myFriendsAddress).transfer(feeAddress, wastedPreMyFriendsTokend);`
Resolution	RESOLVED This typo has been fixed and the contract now sends the correct tokens to the fee address (the actual MyFriends tokens).

Issue #45	**`L2LithSwap`, `preMyFriends`, `preArcadiumAddress`, `myFriendsAddress` and `arcadiumAddress` can be made immutable**
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `L2LithSwap`, `preMyFriends`, `preArcadiumAddress`, `myFriendsAddress` and `arcadiumAddress` explicitly `immutable`.
Resolution	RESOLVED

2.9 AddLiquidityHelper

The AddLiquidityHelper is a simple wrapper contract that can wrap a Uniswap-compliant router like QuickSwap. It simplifies the functions to `add` and `remove` liquidity.

2.9.1 Issues & Recommendations

Issue #46	**`arcadiumAddress` can be changed after initialisation which can lead to siphoning of tokens sent to the contract in case the dev abuses this**
Severity	HIGH SEVERITY
Description	Since the `addLiquidityHelper` is provided as an argument in the ArcadiumToken constructor, the Arcadium token has to be added in manually at a later time. However, the current code implementation allows it to be added multiple times. This allows governance to change the `arcadiumAddress` to a malicious address at a later date which might not be good for investor confidence.
Recommendation	Consider only allowing the `arcadiumAddress` to be set once. This could be done by renouncing ownership within the code itself after this function is called.
Resolution	RESOLVED The Arcadium address can now only be set once.

Issue #47	**Excess tokens will get stuck in the AddLiquidityHelper during liquidity addition**
Severity	LOW SEVERITY
Description	The `addLiquidity` function of a Uniswap router will only add liquidity in proportion to the current pair price. The remaining funds will remain stuck in the AddLiquidityHelper as they are never taken out. Since the amounts to add are provided as function parameters, we consider it unlikely that these will exactly match the current pair price exactly (all though it does seem like the ArcadiumToken closely approximates it).
Recommendation	Consider simply refunding the remaining token balances in the helper to the `msg.sender`. A more advanced resolution is querying the current pair reserves and only taking the required liquidity from the `msg.sender`. This is especially interesting if there is a transfer tax on transfers. The client can take inspiration from Uniswap itself to find an appropriate method to do this.
Resolution	RESOLVED The `addLiquidity` function now transfer out the remaining tokens.

Issue #48	Router is not modifiable while the documentation indicates that it is
Severity	INFORMATIONAL
Description	The AddLiquidityHelper documentation includes the following comment: `// The swap router, modifiable` However, this swap router is not modifiable and only set during contract creation.
Recommendation	Consider whether this router was supposed to be modifiable. Note that if it should be modifiable this would create a new governance issue where the dev can swap out the router for a malicious one. In case it should not be modified directly, consider simply removing the comment.
Resolution	RESOLVED The documentation has been removed.

Issue #49	Functionality is inefficient with transfer-tax tokens
Severity	INFORMATIONAL
Description	Most of the functions will first transfer the tokens into the helper contract, before forwarding them to the Uniswap exchange router. In case either of the two pair tokens has an active transfer-tax, this will result in an inefficiency since two transfers happen.
Recommendation	Ensure that all interactions are excluded from transfer taxes.
Resolution	RESOLVED The client has confirmed that the contracts will be excluded from the native token transfer tax.

Issue #50	The `AddLiquidityHelper` can be used by anyone
Severity	INFORMATIONAL
Description	The `AddLiquidityHelper` functions can be called by anyone. In case this contract is excluded from transfer-taxes, this would mean that anyone can add and remove liquidity without being taxed. We believe this to likely be the intended behavior but have decided to include this issue just to make sure in the unlikely case that this contract is supposed to only be used internally.
Recommendation	Consider whether this is the desired behavior of the contract or whether it should only be usable by a specific set of users and contracts.
Resolution	RESOLVED Client has explained how the LP interactions will be transfer-tax free anyways, so this will not cause a problem.

Issue #51	`arcadiumAddress` and `addArcadiumLiquidity` can be made `immutable`
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `arcadiumAddress` and `addArcadiumLiquidity` explicitly `immutable`.
Resolution	RESOLVED

Issue #52	Lack of events for `setArcadiumAddress`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for `setArcadiumAddress`.
Resolution	RESOLVED

2.10 RHCPToolBox

The RHCPToolBox is a contract that contains a variety of utility functions to calculate the USDC value of any token (using the specified Uniswap router like Quickswap) and to get the current emission schedule.

2.10.1 Privileges

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

- `renounceOwnership`
- `transferOwnership`
- `updateArcadiumSwapRouter`

2.11.2 Issues & Recommendations

Issue #53	`convertToTargetValueFromPair` makes an unnecessary decimal adjustment
Severity	HIGH SEVERITY
Description	In the `convertToTargetValueFromPair` function, a decimal adjustment is made on the resulting number. However, since the reserves already account for the actual decimals these tokens have, no decimal adjustment should be made. Note that there is also a multiplication where a division should occur. But since this logic will have to go we will not turn this in a separate issue.
Recommendation	In case we are misinterpreting the functionality of this function, consider explaining the use-case of it. Otherwise consider carefully testing this function and removing the decimal adjustment functionality.
Resolution	RESOLVED All the decimal adjustment logic has been removed, greatly simplifying the function.

Issue #54	Transactions will revert if the end emission ever reaches zero
Severity	HIGH SEVERITY

Description	In the getArcadiumEmissionForBlock function, the following code can be found:
	`if (endEmission < deltaHeight)`
	`  currentArcadiumEmission = endEmission - deltaHeight;`
	`else`
	`  currentArcadiumEmission = 0;`
	These statements should be the other way around since the subtraction will underflow (causing the transaction to revert) if `endEmission` is smaller than `deltaHeight`.

Recommendation	In case we are misinterpreting the functionality of this function, consider explaining the use-case of it. Otherwise, consider carefully testing this function and removing the decimal adjustment functionality.
Resolution	RESOLVED The typo in the greater-than check has been fixed.

Issue #55	**Wrong emission calculation results in about half of the expected emission rate**
Severity	HIGH SEVERITY
Description	While calculating the reward emissions over a block interval, the following code is used: `(((totalWidth * baseEmission) + (totalWidth * heightDelta) / 1e24) / 2);` The first factor is supposed to capture the base emissions at the start of the interval times the width of that interval. The second factor should capture the average emission rate of the increasing emissions over that period. When looking at it geometrically, the sum of the base emissions should thus indeed be `totalWidth * baseEmission` and then the average of the increased proportion should be `(totalWidth * heightDelta)/2` (assuming a linear increase). However, in the current implementation mentioned above, both factors are divided by two, resulting in approximately half of the desired emission rate.
Recommendation	Consider whether our interpretation is correct and if so, consider adjusting the schedule to something like: `(((totalWidth * baseEmission) + (totalWidth * heightDelta / 2) / 1e24));` in all sections of the code where this miscalculation is made.
Resolution	RESOLVED The logic has been updated as to not divide the first factor by two. The client has indicated that this issue was present due to converting from SafeMath to v0.8.0 native math before the audit (to make the project simpler to audit).

Issue #56	Operator is `private`
Severity	LOW SEVERITY
Description	Currently the operator address is `private`, this makes it difficult for third parties to inspect who the current operator is and that this contract even has governance privileges.
Recommendation	Consider making the operator `public` or simply implementing the resolution of the next issue, which solves this one as well.
Resolution	RESOLVED The operator is removed completely, which makes this contract completely custodian-free.

Issue #57	Operator is `immutable`
Severity	LOW SEVERITY
Description	Currently the operator cannot be passed on to another wallet. This might be an issue if the client ever wants to transfer it to a timelock since this would not be possible.
Recommendation	Consider simply using the OpenZeppelin `Ownable` implementation.
Resolution	RESOLVED The operator is removed completely, which makes this contract completely custodian-free.

Issue #58	**Router is changeable which could cause the contract to break in case it is changed to a malicious contract**
Severity	LOW SEVERITY
Description	Currently the operator can change the router to a malicious one, this could result in the contract returning wrong prices or simply reverting.
Recommendation	In case the router needs to be upgradeable, consider changing the operator to a long timelock so investors can inspect new routers for malicious code.
Resolution	RESOLVED The update router function is removed completely, which makes this contract completely custodian-free.

Issue #59	**`startBlock` can be made `immutable`**
Severity	INFORMATIONAL
Description	Variables that are only set in the constructor but never modified can be indicated as such with the `immutable` keyword. This is considered best practice since it makes the code more accessible for third-party reviewers.
Recommendation	Consider making `startBlock` explicitly `immutable`.
Resolution	RESOLVED

2.11 Timelock

The Timelock contract is a clean fork of Compound Finance's timelock. The only difference is the removal of SafeMath which is not a problem since it is compiled on solidity v0.8.3+commit.8d00100c, which has overflow protection built-in.

2.11.1 Issues & Recommendations

No issues found.

2.12 Locker

The locker is an extremely simple vesting contract which can be used to lock tokens in for a certain period. After the period has expired, the administrator can withdraw any tokens in the contract.

The locker contract will unlock the tokens at block 18723800.

2.12.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY