PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

RimauSwap

04 September 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 RimauSwap 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	RimauSwap
URL	https://rimauswap.finance/
Platform	Binance Smart Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
RimauMasterChef	0xafd546e7bcE1E7d60C7fC7FCf9F101639b09763D	MATCH
RimauToken	0x098dCbf3518856E45BB4e65E7fCc7C5Ff4a2C16e	MATCH
RimauSyrupBar	0x6FE0c579b14D3FfFdCf13c3d4bfEb40AA3675509	MATCH
RimauVault	0x93Aede7f6242DE095E71dd39d338C263bFC58cca	MATCH
SafeMasterchefOwner	0x672875D2DFb5bc4771D09c6E601faFDebb999a36	MATCH
RimauRouter	0x64E8Dad72fEB8Af8261D377A0a2c1441e3589A75	MATCH
RimauFactory	0xA78AAc0C0551ab3470F40ff5A382f0CDbFA31B7b	MATCH
RimauLibrary	Dependency	MATCH
RimauPair	Dependency	MATCH
RimauERC20	Dependency	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	4	3	-	1
Medium	3	0	-	3
Low	8	2	-	6
Informational	23	0	-	23
Total	**38**	**5**	**-**	**33**

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 RimauMasterChef

ID	Severity	Summary	Status
01	HIGH	The migrator function can be used maliciously to steal tokens	RESOLVED
02	HIGH	Severely excessive rewards issue when a token with a transfer tax is added	ACKNOWLEDGED
03	HIGH	Syrup bug is present	RESOLVED
04	MEDIUM	Duplicated pools may be added to the Masterchef	ACKNOWLEDGED
05	MEDIUM	Functions are not secure from reentrancy	ACKNOWLEDGED
06	LOW	Minting will break if multiplier is set to a very high value	ACKNOWLEDGED
07	LOW	Adding an EOA or a non-token contract as a pool will break `updatePool` and `massUpdatePools`	ACKNOWLEDGED
08	LOW	The `pendingCake` function will revert if `totalAllocPoint` is zero	ACKNOWLEDGED
09	INFO	`cake` and `syrup` can be made immutable	ACKNOWLEDGED
10	INFO	Pools use the contract balance to figure out the total deposits	ACKNOWLEDGED
11	INFO	Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions	ACKNOWLEDGED
12	INFO	`dev` function can be renamed	ACKNOWLEDGED
13	INFO	`updateMultiplier`, `add`, `set`, `deposit`, `withdraw`, `enterStaking`, `leaveStaking`, `emergencyWithdraw` and `dev` functions can be made `external`	ACKNOWLEDGED
14	INFO	Lack of events for `updateMultiplier`, `add`, `set`, `dev` and `emergencyWithdraw`	ACKNOWLEDGED

1.3.2 RimauToken

ID	Severity	Summary	Status
15	LOW	mint function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	RESOLVED
16	INFO	Governance functionality is broken	ACKNOWLEDGED
17	INFO	`delegateBySig` can be frontrun and cause denial of service	ACKNOWLEDGED
18	INFO	Contract still references Cake and Rimau is mis-spelled	ACKNOWLEDGED

1.3.3 RimauSyrupBar

ID	Severity	Summary	Status
19	LOW	`mint` function can be used to pre-mint large amounts of Syrup tokens before ownership is transferred to the Masterchef	RESOLVED
20	INFO	Governance functionality is broken	ACKNOWLEDGED
21	INFO	`delegateBySig` can be frontrun and cause denial of service	ACKNOWLEDGED
22	INFO	Contract still references Cake	ACKNOWLEDGED

1.3.4 RimauVault

ID	Severity	Summary	Status
23	HIGH	Admin could take out all tokens by calling emergencyWithdraw and harvest repeatedly	RESOLVED
24	INFO	Contract cannot handle tokens with transfer taxes	ACKNOWLEDGED
25	INFO	pause and unpause emit two events	ACKNOWLEDGED
26	INFO	Contract still references Cake	ACKNOWLEDGED
27	INFO	Lack of events for `setAdmin`, `setTreasury`, `setPerformanceFee`, `setCallFee`, `setWithdrawFee`, `setWithdrawFeePeriod`, `emergencyWithdraw` and `inCaseTokensGetStuck`	ACKNOWLEDGED

1.3.5 SafeMasterchefOwner

ID	Severity	Summary	Status
28	LOW	Adding EOA or non-token contract as a pool is possible	ACKNOWLEDGED
29	LOW	`updateMultiplier` has no safeguards	ACKNOWLEDGED
30	INFO	`updateMultiplier`, `add` and `set` functions can be made `external`	ACKNOWLEDGED
31	INFO	Lack of events for `updateMultiplier`, `add` and `set`	ACKNOWLEDGED
32	INFO	`devAddr` cannot be changed	ACKNOWLEDGED

1.3.6 RimauRouter

ID	Severity	Summary	Status
33	INFO	Phishing is possible by a malicious frontend by adjusting routes, tokens or `from` parameters (Also present in Uniswap and Pancakeswap)	ACKNOWLEDGED

1.3.7 RimauFactory

No issues found.

1.3.8 RimauLibrary

ID	Severity	Summary	Status
34	MEDIUM	Fee is miscalculated in `getAmountOut` and `getAmountIn` functions causing users to pay a slightly excessive fee of 0.05% when they trade	ACKNOWLEDGED

1.3.9 RimauPair

ID	Severity	Summary	Status
35	INFO	Pairs without supply but with a partial reserve might crash the frontend if the user wants to swap on this pair (this issue is present in most frontends)	ACKNOWLEDGED

1.3.10 RimauERC20

ID	Severity	Summary	Status
36	LOW	Approval event is not emitted if allowance is changed in `transferFrom` as suggested in the ERC-20 Token Standard (also present in Uniswap)	ACKNOWLEDGED
37	INFO	Permit can be front-run to prevent someone from calling `removeLiquidityWithPermit` (also present in Uniswap)	ACKNOWLEDGED
38	INFO	Redundant comments can be deleted	ACKNOWLEDGED

1.3.11 Helper Libraries

No issues found.

2 Findings

2.1 RimauMasterChef

The RimauMasterChef is a fork of Pancakeswap's Masterchef with the migrator code intact. There is a risk of this being used maliciously to siphon out user funds at any time, as was the case with several protocols in the past that did contain the same migrator function. There are no deposit fees. The syrup bug is present (read more here).

2.1.1 Privileged Roles

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

- `add`
- `set`
- `setMigrator`
- `migrate`
- `dev`

2.1.2 Issues & Recommendations

Issue #01	The migrator function can be used maliciously to steal tokens
Severity	HIGH SEVERITY
Description	This function can be used to steal all staked assets in the Masterchef, and poses a significant risk of total loss to users.
Recommendation	It is recommended to entirely remove the `migrate` function. The Pancakeswap team themselves have stated here that this migrator code is destructive and should not be called.
Resolution	RESOLVED The client has stated that they have utilized the SafeMasterchefOwner wrapper contract to mitigate the migrator function.

Issue #02	Severely excessive rewards issue when a token with a transfer tax is added
Severity	HIGH SEVERITY
Description	When tokens with a transfer tax are added to the pools, this will result in significant excessive rewards. Due to the way the Masterchef handles rewards, rewards can be heavily inflated when the balance of the Masterchef no longer matches that of user deposits. This happens for example with transfer tax tokens. This issue is further amplified on Masterchefs like this one with a referral mechanism, since tokens can be minted directly. This flaw of the Masterchef has recently been exploited on a significant number of projects, all of which their native tokens went to $0 afterwards because the exploit resulted in a large number of native tokens being minted and dumped. This issue was also present in SushiSwap (the original Masterchef). Since they were never meant to have any tokens but LP tokens, it was not a problem there but has become a problem to projects who have started forking it for usage with less standard tokens.
Recommendation	Consider using the current standard of handling deposits, which is based on how Uniswap handles transfer fees:

```
uint256 balanceBefore = pool.lpToken.balanceOf(address(this));
pool.lpToken.transferFrom(msg.sender, address(this), _amount);
_amount = pool.lpToken.balanceOf(address(this)).sub(balanceBefore);
```

Resolution	ACKNOWLEDGED

Issue #03	Syrup bug is present
Severity	HIGH SEVERITY
Description	The Syrup tokens are minted when users stake in the first pool, and burned when the `withdraw` function is called. Unfortunately, the Syrup tokens are not burned when `emergencyWithdraw` is called, resulting in users being able to exploit the issue and inflate their Syrup token balance – in Pancakeswap's case, to the tune of 30 million Syrup tokens.

Recommendation	Consider burning syrup tokens in the `emergencyWithdraw` function, like so:
	
	`function emergencyWithdraw(uint256 _pid) external nonReentrant {`
	`  PoolInfo storage pool = poolInfo[_pid];`
	`  UserInfo storage user = userInfo[_pid][msg.sender];`
	`  uint256 amount = user.amount;`
	`  user.amount = 0;`
	`  user.rewardDebt = 0;`
	`  if(_pid == 0) {`
	`  syrup.burn(msg.sender, amount);`
	`  }`
	`  pool.lpToken.safeTransfer(address(msg.sender), amount);`
	`  emit EmergencyWithdraw(msg.sender, _pid, amount);`
	`}`

Resolution	RESOLVED The client has stated that they will not be using the syrup tokens.

Issue #04	Duplicated pools may be added to the Masterchef
Severity	MEDIUM SEVERITY
Description	The add function allows for duplicate pools to be added, which would lead to dilution of emission rewards to stakers.

Recommendation	The addition of a modifier that checks for duplicate pools could help prevent this from occurring.
	
	`mapping(IBEP20 => bool) public poolExistence;`
	`modifier nonDuplicated(IBEP20 _lpToken) {`
	`  require(poolExistence[_lpToken] == false, "nonDuplicated: duplicated");`
	`  _;`
	`}`
	
	Alternatively, you could account for this by adding in an `lpSupply` variable under `poolInfo`. This has the benefit of accounting for accurately accounting for deposits in the Masterchef.

Resolution	ACKNOWLEDGED

Issue #05	Functions are not secure from reentrancy
Severity	MEDIUM SEVERITY
Description	All deposit, withdraw and staking-related functions are susceptible to reentrancy, especially if ERC777 tokens are added to the contract. 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 the `nonReentrant` modifier to all `deposit`, `withdraw`, `emergencyWithdraw`, `enterStaking` and `leaveStaking` functions, and reordering line items in the functions to ensure best safety practices are adhered to per the Check Effects Interactions pattern.
Resolution	ACKNOWLEDGED

Issue #06	Minting will break if multiplier is set to a very high value
Severity	LOW SEVERITY
Description	If the multiplier is set to an extremely large value using `updateMultiplier`, SafeMath will make the `updatePool` calls fail due to overflow. This will break any minting, depositing and withdrawing.
Recommendation	Consider adding a safeguard to the updateMultiplier function with a reasonable limit: `require(multiplierNumber < MAX_MULTIPLIER);`
Resolution	ACKNOWLEDGED

Issue #07	Adding an EOA or a non-token contract as a pool will break `updatePool` and `massUpdatePools`
Severity	LOW SEVERITY
Description	`updatePool` will always call `balanceOf(address(this))` on the token of this pool, and will fail if the token is not an actual token contract address.
Recommendation	Consider simply adding a test line in the add function. If the token does not exist, this will make sure the add function fails. `_lpToken.balanceOf(address(this));`
Resolution	ACKNOWLEDGED

Issue #08	The `pendingCake` function will revert if `totalAllocPoint` is zero
Severity	LOW SEVERITY
Description	In the `pendingCake` function, at some point a division is made by the `totalAllocPoint` variable. If all pools have their rewards set to zero, this variable will be zero as well. The requests will then revert with a division by zero error.
Recommendation	Consider only calculating the accumulated rewards since the `lastRewardBlock` if the `totalAllocPoint` variable is greater than zero. This check can simply be added to the existing check that verifies the `block.number` and `lpSupply`, like so: `if (block.number > pool.lastRewardBlock && lpSupply != 0 && totalAllocPoint > 0) {`
Resolution	ACKNOWLEDGED

Issue #09	cake and syrup 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 `cake` (this should be changed to the native token name) and `syrup` explicitly `immutable`.
Resolution	ACKNOWLEDGED

Issue #10	Pools use the contract balance to figure out the total deposits
Severity	INFORMATIONAL
Description	As with pretty much all Masterchefs, the total number of tokens in the Masterchef contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the Masterchef. More severely, because the native token is constantly minted, this will cause severe dilution on the native token pool.
Recommendation	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits. Each `lpToken.balanceOf(address(this))` query can then be replaced with this `lpSupply` as well.
Resolution	ACKNOWLEDGED

Issue #11	**Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions**
Severity	INFORMATIONAL
Description	Within `updatePool`, `accCakePerShare` is based on the `lpSupply` variable. `pool.accCakePerShare = pool.accCakePerShare.add(cakeReward.mul(1e12).div(lpSupply));` However, if this `lpSupply` becomes a severely large value, precision errors may occur due to rounding. This is famously seen when pools decide to add meme-tokens which usually have huge supplies and no decimals.
Recommendation	Consider increasing precision to `1e18` across the entire contract.
Resolution	ACKNOWLEDGED

Issue #12	**dev function can be renamed**
Severity	INFORMATIONAL
Description	Users may be confused about what the `dev` function does.
Recommendation	Consider renaming this function to `setDevAddress`.
Resolution	ACKNOWLEDGED

Issue #13	**`updateMultiplier`, `add`, `set`, `deposit`, `withdraw`, `enterStaking`, `leaveStaking`, `emergencyWithdraw` and dev 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	ACKNOWLEDGED

Issue #14	**Lack of events for `updateMultiplier`, `add`, `set`, dev and `emergencyWithdraw`**
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	ACKNOWLEDGED

2.2 RimauToken

The RimauToken contract allows for Rimau tokens to be minted when the `mint` function is called by Owner, who at the time of deployment would be the deployer. Ownership is generally transferred to the Masterchef via the `transferOwnership` function to allow for emission rewards to be minted and distributed to users staking in the Masterchef. The `mint` function can be used to pre-mint tokens for various uses including injection of initial liquidity, token presale, airdrops, and others.

2.2.1 Token Overview

Address	`0x098dCbf3518856E45BB4e65E7fCc7C5Ff4a2C16e`
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None
Pre-mints	1 million

2.2.2 Privileged Roles

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

- `mint`

2.2.3 Issues & Recommendations

Issue #15	**`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; however, this function may also be used to pre-mint and dump tokens when the token contract has been deployed but before ownership is set to the Masterchef contract. This risk is prevalent amongst less-reputable projects, and any pre-mints can be prominently seen on the Blockchain.
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 1 million tokens were pre-minted and token ownership has been transferred to the Masterchef in the latest deployed contracts provided to us.

Issue #16	Governance functionality is broken
Severity	INFORMATIONAL
Description	Although there is YAM-related delegation code in the token contract which is usually used for governance and voting, the delegation code can be abused as the delegates are not moved during transfers and burns. This allows for double spending attacks on the voting mechanism. It should be noted that this issue is present in pretty much every single farm out there including PancakeSwap and even SushiSwap.
Recommendation	The broken delegation-related code can be removed to reduce the size of the contract. If voting is ever desired, it can still be done through snapshot.org, used by many of the larger projects.
Resolution	ACKNOWLEDGED

Issue #17	`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	Similar to the issue before this, the code can just be removed.
Resolution	ACKNOWLEDGED

Issue #18	Contract still references Cake and Rimau is mis-spelled
Severity	INFORMATIONAL

Description	Throughout the contract, the token, comments and functions all still reference Cake. This is because the contract was forked from Pancakeswap. Rimau is also spelled wrongly. A non-exhaustive list is presented below:
	
	Line 862
	`// CakeToken with Governance.`
	
	Line 863 (Typo in Rimau?)
	`contract RIMAUToken is BEP20('Ximau', 'Ximau') {`

Recommendation	To make the contract more consistent and thus readable for third-party reviewers, consider replacing all mentions of Cake with the protocol's native token, Rimau. Consider also renaming Ximau to Rimau.
Resolution	ACKNOWLEDGED

2.3 RimauSyrupBar

The RimauSyrupBar contract mints syrup tokens when users stake into the first pool in the Masterchef via `enterStaking`, and burns them when users withdraw their funds via `leaveStaking`.

2.3.1 Token Overview

Address	0x6FE0c579b14D3FfFdCf13c3d4bfEb40AA3675509
Token Supply	Unlimited
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:

- `mint`

2.3.3 Issues & Recommendations

Issue #19	`mint` function can be used to pre-mint large amounts of Syrup tokens before ownership is transferred to the Masterchef
Severity	LOW SEVERITY
Description	Prior to ownership of the SyrupBar contract being transferred to the Masterchef, there is a risk that the contract deployer minting any amount of Syrup tokens.
Recommendation	Consider immediately transferring ownership of the SyrupBar contract to the Masterchef. Once this has been done, we can mark the issue as Resolved assuming no Syrup tokens were pre-minted.
Resolution	RESOLVED The client has stated that they will not be using the Syrup contract.

Issue #20	Governance functionality is broken
Severity	INFORMATIONAL
Description	Although there is YAM-related delegation code in the token contract which is usually used for governance and voting, the delegation code can be abused as the delegates are not moved during transfers and burns. This allows for double spending attacks on the voting mechanism. It should be noted that this issue is present in pretty much every single farm out there including PancakeSwap and even SushiSwap.
Recommendation	The broken delegation-related code can be removed to reduce the size of the contract. If voting is ever desired, it can still be done through snapshot.org, used by many of the larger projects.
Resolution	ACKNOWLEDGED

Issue #21	`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	Similar to the issue before this, the code can just be removed.
Resolution	ACKNOWLEDGED

Issue #22	Contract still references Cake
Severity	INFORMATIONAL

Description	Throughout the contract, the token, comments and functions all still reference Cake. This is because the contract was forked from Pancakeswap. A non-exhaustive list is presented below:
	
	Line 1114
	`RimauToken public cake;`
	
	Line 1118
	`RimauToken _cake`
	
	Line 1124
	`function safeCakeTransfer(address _to, uint256 _amount) public onlyOwner`

Recommendation	To make the contract more consistent and thus readable for third-party reviewers, consider replacing all mentions of Cake with the protocol's native token, Rimau.
Resolution	ACKNOWLEDGED

2.4 RimauVault

The RimauVault is a standard native token compounding vault identical to the one in Pancakeswap. However, since Paladin was also commissioned to audit this contract, we carried out a careful examination of this contract. The contract deposits into the underlying Masterchef via the staking method, which is reserved for the native Rimau token only.

The contract has a performance fee of 2% (up to a maximum of 5%), a call fee of 0.25% (up to a maximum of 1%), a withdraw fee of 0.1% (up to a maximum of 1%) which is sent to the treasury and a withdraw fee period of 3 days (up to a maximum of 3 days). This means that after this period, a fee will no longer be deducted from the staked amount upon withdrawal.

The `inCaseTokensGetStuck` function allows the contract admin to withdraw tokens sent to the contract by mistake. Note that this function cannot be called on the native or receipt tokens.

2.4.1 Privileged Roles

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

- `setAdmin`
- `setTreasury`
- `inCaseTokensGetStuck`
- `emergencyWithdraw`
- `setWithdrawFeePeriod`
- `setWithdrawFee`
- `setCallFee`
- `setPerformanceFee`

2.4.2 Issues & Recommendations

Issue #23	**Admin could take out all tokens by calling `emergencyWithdraw` and `harvest` repeatedly**
Severity	HIGH SEVERITY
Description	The contract is vulnerable to a centralisation issue that allows the admin to withdraw all native tokens by calling `emergencyWithdraw` and `harvest` repeatedly. This exploit works due to the fact that `harvest` will subtract a performance fee from the vault balance as it assumes that that balance was created from rewards. However, due to the `emergencyWithdraw` function, this balance can be the whole vault balance. By repeating this sequence many times, the admin can effectively drain the vault.
Recommendation	In the long term, consider fundamentally resolving this function by not allowing this sequence. In the short run, consider moving the admin to a sufficiently long Timelock so all investors can react in due time.
Resolution	RESOLVED The client has removed the `emergencyWithdraw` function.

Issue #24	Contract cannot handle tokens with transfer taxes
Severity	INFORMATIONAL
Description	If this contract is ever forked for use with a token that has a transfer tax (think a second layer), this contract will not function correctly since the deposit function does not account for transfer taxes, and thus be could be exploited like other Masterchefs that have the same bug with regards to tokens with transfer taxes. Although this is not an issue in the current deployment, we believe it could be useful for the team to know this for their future endeavours.
Recommendation	No resolution required. This issue can be marked as Resolved once the client has acknowledged this.
Resolution	ACKNOWLEDGED

Issue #25	pause and unpause emit two events
Severity	INFORMATIONAL
Description	The pause and unpause method emit an event both at the vault level and at the library level. It is not really necessary to emit two events.
Recommendation	Consider removing the events at the vault level and relying on the ones at the Pausable library level.
Resolution	ACKNOWLEDGED

Issue #26	Contract still references Cake
Severity	INFORMATIONAL

Description	Throughout the contract, the token, comments and functions still reference Cake. This is because the contract was forked from Pancakeswap. A non-exhaustive list is presented below:
	
	Line 846
	`// Cake token`
	
	Line 876
	`* @param _token: Cake token contract`
	
	Line 1073
	`function calculateHarvestCakeRewards() external view returns (uint256) {`

Recommendation	To make the contract more consistent and thus readable for third-party reviewers, consider replacing all mentions of Cake with the protocol's native token, Rimau.
Resolution	ACKNOWLEDGED

Issue #27	Lack of events for `setAdmin`, `setTreasury`, `setPerformanceFee`, `setCallFee`, `setWithdrawFee`, `setWithdrawFeePeriod`, `emergencyWithdraw` and `inCaseTokensGetStuck`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above functions.
Resolution	ACKNOWLEDGED

2.5 SafeMasterchefOwner

This is a wrapper contract to which ownership of the Masterchef contract will be transferred to in order to mitigate the high-risk issue of the migrator. Once ownership of the Masterchef has been transferred to this contract, it cannot be reversed. The wrapper contract can only call three functions: `add`, `set` and `updateMultiplier` in the underlying Masterchef contract.

We highly recommend that the migrator should be removed at the source in the Masterchef contract itself rather than relying on a wrapper contract post-deployment, as the presence of the migrator poses a significant risk to users until the wrapper contract is utilized.

2.5.1 Privileged Roles

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

- `add`
- `set`
- `updateMultiplier`

2.5.2 Issues & Recommendations

Issue #28	Adding EOA or non-token contract as a pool is possible
Severity	LOW SEVERITY
Description	`updatePool` will always call `balanceOf(address(this))` on the token of this pool in the Masterchef, and will fail to do so if it is not a token contract.
Recommendation	Consider simply adding a test line in the add function. If the token does not exist, this will make sure the add function fails. `_lpToken.balanceOf(address(this));`
Resolution	ACKNOWLEDGED

Issue #29	`updateMultiplier` has no safeguards
Severity	LOW SEVERITY
Description	If the multiplier is set to an extremely large value using `updateMultiplier`, SafeMath will make the `updatePool` calls fail due to overflow. This will break any minting, depositing and withdrawing.
Recommendation	Consider adding a safeguard to the `updateMultiplier` function with a reasonable limit: `require(multiplierNumber < MAX_MULTIPLIER);`
Resolution	ACKNOWLEDGED

Issue #30	`updateMultiplier`, `add` and `set` functions can be made `external`
Severity	INFORMATIONAL
Description	The `updateMultiplier`, `add`, `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.
Recommendation	Consider making these functions `external`.
Resolution	ACKNOWLEDGED

Issue #31	Lack of events for `updateMultiplier`, `add` and `set`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for these variables.
Resolution	ACKNOWLEDGED

Issue #32	`devAddr` cannot be changed
Severity	INFORMATIONAL
Description	The wrapper contract does not allow for the `devAddr` to be changed in the underlying Masterchef, if there should ever be a need to do so.
Recommendation	No resolutions are required. This is just informational.
Resolution	ACKNOWLEDGED

2.6 RimauRouter

The RimauSwap AMM protocol, forked with minimal changes from PancakeSwap V2, uses the RimauRouter as an entry point for users to exchange tokens. The RimauRouter is responsible for determining the swap rate and allowing for user interactions to be done with safety checks. More specifically, the RimauRouter allows users to add liquidity, remove liquidity and swap tokens.

2.6.1 Issues & Recommendations

Issue #33	**Phishing is possible by a malicious frontend by adjusting routes, tokens or `from` parameters (Also present in Uniswap and Pancakeswap)**
Severity	INFORMATIONAL
Description	A malicious (for example hacked) frontend can easily mislead users in approving malicious transactions, even if the router matches the address described in this report. An obvious example of how this can be done is by changing the `to` parameter which indicates to whom tokens or liquidity has to be sent. Other ways to phish could include using malicious routes or tokens.
Recommendation	Consider carefully protecting the frontend and ideally having an unchangeable IPFS fallback implementation for it.
Resolution	ACKNOWLEDGED

2.7 RimauFactory

The RimauFactory is the management contract for the RimauSwap AMM – it keeps track of all RimauPairs and allows users to create new ones. Any RimauPair created through the verified factory is immediately verified as well, since the pair is deployed by the verified factory. It is a clean fork of PancakeSwap V1 with minimal changes.

2.7.1 Issues & Recommendations

No issues found.

2.8 RimauLibrary

The RimauLibrary contract is a dependency contract used to calculate the appropriate trading rates. It is used by the RimauRouter to calculate how many tokens should be sent to the pairs and is thus an important component of the user-facing aspect of the system.

2.8.1 Issues & Recommendations

Issue #34	**Fee is miscalculated in `getAmountOut` and `getAmountIn` functions causing users to pay a slightly excessive fee of 0.05% when they trade**
Severity	MEDIUM SEVERITY

Location	Lines 323-330
	`function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) internal pure returns (uint amountOut) {`
	`  require(amountIn > 0, 'RimauLibrary: INSUFFICIENT_INPUT_AMOUNT');`
	`  require(reserveIn > 0 && reserveOut > 0, 'RimauLibrary: INSUFFICIENT_LIQUIDITY');`
	`  uint amountInWithFee = amountIn.mul(9975);`
	`  uint numerator = amountInWithFee.mul(reserveOut);`
	`  uint denominator = reserveIn.mul(10000).add(amountInWithFee);`
	`  amountOut = numerator / denominator;`
	`}`
	
	Lines 333-339
	`function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) internal pure returns (uint amountIn) {`
	`  require(amountOut > 0, 'RimauLibrary: INSUFFICIENT_OUTPUT_AMOUNT');`
	`  require(reserveIn > 0 && reserveOut > 0, 'RimauLibrary: INSUFFICIENT_LIQUIDITY');`
	`  uint numerator = reserveIn.mul(amountOut).mul(10000);`
	`  uint denominator = reserveOut.sub(amountOut).mul(9975);`
	`  amountIn = (numerator / denominator).add(1);`
	`}`

Description	Within the RimauLibrary, the trading fee reduction has not been adjusted to the lowered fee of 0.2% in RimauPair, which may have been caused by the mix-and-match of forking PancakeSwap's V2 Router and pairing that with PancakeSwap's V1 Factory. This results in the router sending excessive tokens to the pairs and the users trading at a slightly worse rate then they are allowed to trade at.
Recommendation	Consider modifying `mul(9975)` to `mul(9980)` to account for the 0.2% fee used in RimauPair.
Resolution	ACKNOWLEDGED

2.9 RimauPair

The RimauPair is the core component of the RimauSwap AMM protocol, it represents a pair of two tokens. People can add liquidity in this pair by depositing both tokens in equally valued proportion for others to swap against. It is a clean fork from PancakeSwap V1.

2.9.1 Issues & Recommendations

Issue #35	**Pairs without supply but with a partial reserve might crash the frontend if the user wants to swap on this pair (this issue is present in most frontends)**
Severity	INFORMATIONAL
Description	A malicious DoS attack we have witnessed in practice is when a project wants to go live through a presale, people can instantiate the pair while there are no tokens yet. The malicious party will then sent some of the counterparty token to this pair so it has a partial balance (eg. 0.1 BNB and 0 tokens). When `sync()` is then called, the pairs' reserves are updated to account for this balance. Due to a division by zero exception, many frontends can not properly account for this state and will go through a blank page, preventing the original project from adding liquidity through the frontend.
Recommendation	Consider checking whether this is present in the frontend and adding a division by zero handler.
Resolution	ACKNOWLEDGED

2.10 RimauERC20

The RimauERC20 is an implementation of the ERC-20 Token Standard. It is a clean copy of the related Pancakeswap contract.

2.10.1 Issues & Recommendations

Issue #36	**Approval event is not emitted if allowance is changed in `transferFrom` as suggested in the ERC-20 Token Standard (also present in Uniswap)**
Severity	LOW SEVERITY
Location	Lines 179-185 `function transferFrom(address from, address to, uint value) external returns (bool) { if (allowance[from][msg.sender] != uint(-1)) { allowance[from][msg.sender] = allowance[from][msg.sender].sub(value); } _transfer(from, to, value); return true; }`
Description	The ERC-20 standard specifies that an approval event should be emitted when the allowance of a user changes. However, within the ERC20 implementation of both Uniswap and the RimauSwap, this is not done. You can read more about this improvement in Pull Request #65 of uniswap-core.
Recommendation	Consider adding `emit Approval(from, msg.sender, remaining)` in `transferFrom` when allowance is modified.
Resolution	ACKNOWLEDGED

Issue #37	**Permit can be front-run to prevent someone from calling `removeLiquidityWithPermit` (also present in Uniswap)**
Severity	INFORMATIONAL

Location	Lines 187-199
	`function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external {`
	`    require(deadline >= block.timestamp, 'Rimau: EXPIRED');`
	`    bytes32 digest = keccak256(`
	`        abi.encodePacked(`
	`            '\x19\x01',`
	`            DOMAIN_SEPARATOR,`
	`            keccak256(abi.encode(PERMIT_TYPEHASH, owner, spender, value, nonces[owner]++, deadline))`
	`        )`
	`    );`
	`    address recoveredAddress = ecrecover(digest, v, r, s);`
	`    require(recoveredAddress != address(0) && recoveredAddress == owner, 'Rimau: INVALID_SIGNATURE');`
	`    _approve(owner, spender, value);`
	`}`

Description	Currently if `permit` is executed twice, the second execution will be reverted. It is thus in theory possible for a bot to pick up permit transactions in the mempool and execute them before a contract can. The implications of this issue is that a bad actor could prevent a user from removing liquidity with a permit through the router. It is a denial of service attack which is present in all AMMs but which we have yet to witness being used since there is no profit from it.
Recommendation	Consider this issue if there are ever complaints by users that their `removeLiquidityWithPermit` transactions are failing. It could be the case that someone is using this vector against them. We do not recommend changing this behavior since it would cause a lot of extra work modifying the frontend to account for new permit behavior. This issue is also present in Uniswap after all.
Resolution	ACKNOWLEDGED

Issue #38	Redundant comments can be deleted
Severity	INFORMATIONAL

Location	Line 122
	`// TODO: to be check wheather need to calculate or not - dilip`
	
	Line 261
	`// TODO: Check this pancakeCall - dilip - change`

Description	There are redundant comments within the code.
Recommendation	These comments can be deleted to reduce the length of the contract.
Resolution	ACKNOWLEDGED

2.11 Helper Libraries

2.11.1 Math, SafeMath, TransferHelper, UQ112x112

Math, SafeMath, TransferHelper and UQ112x112 are various helper libraries which are each identical to the PancakeSwap implementation.

2.11.2 Issues & Resolutions

No issues found.


PALADIN
BLOCKCHAIN SECURITY