PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Kavian Finance

22 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 the Kavian Finance on the Polygon network. 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	Kavian Finance
URL	https://kavian.finance/
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`MasterChef`	`0xB664c98548CEbf7024F899e32E467dff00311918`	MATCH
`KavianL2`	`0x9A33BAC266b02fAfF8fa566C8Cb5da08820E28ba`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	7	-	-	7
Informational	13	-	-	13
Total	**20**	**0**	**0**	**20**

Classification of Issues

Severity	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 MasterChef

ID	Severity	Summary	Status
01	LOW	Duplicated pools may be added to the Masterchef	ACKNOWLEDGED
02	LOW	The pendingKavianL2 function will revert if totalAllocPoint is zero	ACKNOWLEDGED
03	LOW	Adding an EOA or a non-token contract as a pool will break updatePool and massUpdatePools	ACKNOWLEDGED
04	LOW	updateEmissionRate has no maximum safeguard	ACKNOWLEDGED
05	LOW	There are no sanity checks in the `setKavianL2Referral` function	ACKNOWLEDGED
06	INFO	`BONUS_MULTIPLIER` is redundant	ACKNOWLEDGED
07	INFO	Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions	ACKNOWLEDGED
08	INFO	`kavianl2` can be made immutable	ACKNOWLEDGED
09	INFO	Pools use the contract balance to figure out the total deposits	ACKNOWLEDGED
10	INFO	add, set, deposit, withdraw, emergencyWithdraw, setDevAddress, setFeeAddress, setTeamAddress, updateEmissionRate, setKavianL2Referral and setReferralCommission functions can be made external	ACKNOWLEDGED
11	INFO	Lack of events for add, set, setDevAddress, setFeeAddress, setTeamAddress, setKavianL2Referral, setReferralCommission and setStartBlock	ACKNOWLEDGED

1.3.2 KavianL2

ID	Severity	Summary	Status
12	LOW	The `updateMaxTransferAmountRate` minimum safeguard is too low	ACKNOWLEDGED
13	LOW	LP tokens are sent to the operator address which could be an EOA	ACKNOWLEDGED
14	LOW	Contract uses raw addition	ACKNOWLEDGED
15	INFO	Incorrect comment	ACKNOWLEDGED
16	INFO	Excluding zero address can be removed	ACKNOWLEDGED
17	INFO	Governance functionality is broken	ACKNOWLEDGED
18	INFO	`delegateBySig` can be frontrun and cause denial of service	ACKNOWLEDGED
19	INFO	Lack of events for `setExcludedFromAntiWhale`, `setLiquidityAddress` and `setSwapPair`	ACKNOWLEDGED
20	INFO	Functions can be made external	ACKNOWLEDGED

2 Findings

2.1 MasterChef

The Kavian Masterchef is a modified Panther Masterchef fork. Just like Panther, rewards can only be harvested after a specified interval (configurable to a maximum of 14 days) has passed. The referral commission rate is initially set to 1% and can be set to a maximum of 10%. The deposit fees can be split over two addresses: `feeAddress` and `teamAddress`. Finally, deposit fees can be set to a maximum value of 10%.

The Masterchef mints 10% of generated rewards to the `devAddress`, and a further 100% to stakers, for a cumulative total of 110% minting (note that this is very common amongst most Masterchefs).

2.1.1 Privileged Roles

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

- `add`
- `set`
- `setDevAddress`
- `setFeeAddress`
- `setTeamAddress`
- `updateEmissionRate`
- `setKavianL2Referral`
- `setReferralCommissionRate`
- `setStartBlock`

2.1.2 Issues & Recommendations

Issue #01	Duplicated pools may be added to the Masterchef
Severity	LOW 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 incident 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 #02	**The `pendingKavianL2` function will revert if `totalAllocPoint` is zero**
Severity	LOW SEVERITY
Description	In the `pendingKavianL2` 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 #03	**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(s)	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 #04	`updateEmissionRate` has no maximum safeguard
Severity	LOW SEVERITY
Description	Projects sometimes accidentally update their emission rate to a severely high number either by accident or with malicious intent.
Recommendation	Consider adding a `MAX_EMISSION_RATE` variable and setting it to a reasonable value. `require(_kavianL2PerBlock <= MAX_EMISSION_RATE,"Too high");`
Resolution	ACKNOWLEDGED

Issue #05	There are no sanity checks in the `setKavianL2Referral` function
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.
Recommendation	Consider making the referral address non-upgradeable (only settable once) to ensure that functionality can never break, such as setting it in the constructor. We rarely ever see a project updating their referral after it is initially set.
Resolution	ACKNOWLEDGED

Issue #06	**BONUS_MULTIPLIER is redundant**
Severity	INFORMATIONAL
Description	The constant variable `BONUS_MULTIPLIER` does not look to be used in the Masterchef contract and can thus be removed.
Recommendation	Consider removing `BONUS_MULTIPLIER` and associated comments.
Resolution	ACKNOWLEDGED

Issue #07	**Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions**
Severity	INFORMATIONAL
Description	Within `updatePool`, `accMorphPerShare` is based on the `lpSupply` variable. `pool.accMorphPerShare = pool.accMorphPerShare.add(morphReward.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 #08	`kavianl2` 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 and saves gas.
Recommendation	Consider making `kavianl2` explicitly `immutable`.
Resolution	ACKNOWLEDGED

Issue #09	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 #10	**add, set, deposit, withdraw, emergencyWithdraw, setDevAddress, setFeeAddress, setTeamAddress, updateEmissionRate, setKavianL2Referral and setReferralCommission 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 #11	**Lack of events for add, set, setDevAddress, setFeeAddress, setTeamAddress, setKavianL2Referral, setReferralCommission and setStartBlock**
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 KavianL2

The KavianL2 token is a simple Panther token fork, with anti-whale currently set at 8% (can be set to a minimum of 0.5%) of total supply, and transfer taxes of 5% (can be set to a maximum of 10%), of which one-fifths is burned. This contract would periodically add Kavian LP using tokens that accumulate in the balance, and the proceeds would go to the operator address, though this can be modified to a different address at any time in the future.

2.2.1 Token Overview

Address	0x9A33BAC266b02fAfF8fa566C8Cb5da08820E28ba
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	100%
Transfer Min Size	0.5%
Transfer Fees	Up to 10%

2.2.2 Privileged Roles

The following are functions of the KavianL2 that is callable by the owner:

- `updateTransferTaxRate`
- `updateBurnRate`
- `updateMaxTransferAmountRate`
- `updateMinAmountToLiquify`
- `setExcludedFromAntiWhale`
- `updateSwapAndLiquifyEnabled`
- `setLiquidityAddress`
- `setSwapPair`

2.2.3 Issues & Recommendations

Issue #12	**The `updateMaxTransferAmountRate` minimum safeguard is too low**
Severity	LOW SEVERITY
Description	The purpose of anti-whales is to prevent large amounts of selling quickly. However, setting the minimum to only 0.5% allows the project to hamper users' efforts in trying to liquidate their positions in times of need or haste. Additionally, setting it so low only marginally delays the selling pressure, and would likely cause further chaos and anger should there be any marked attempt to prevent users from exercising their freedom to liquidate.
Recommendation	Consider increasing the minimum to at least 1-2%, which would be a more tolerable amount for many users. Third-party reviewers such as RugDoc generally advocate for a minimum of 1-2% as well. `require(_maxTransferAmountRate >= 200, "KAVIANL2::updateMaxTransferAmountRate: Max transfer amount rate must be grater than min rate");`
Resolution	ACKNOWLEDGED

Issue #13	LP tokens are sent to the operator address which could be an EOA
Severity	LOW SEVERITY
Description	Part of the transaction fee is used to generate LP tokens. These are sent to the operator address (though this can be changed to any address), which is normally an EOA. As a result, these may be offloaded or dumped on users.
Recommendation	Consider sending the LP tokens to a vesting contract which will release the tokens gradually over time. Alternatively, sending them to the burn address may be feasible and may be looked upon favourably by users.
Resolution	ACKNOWLEDGED

Issue #14	Contract uses raw addition
Severity	LOW SEVERITY

Location	Line 131
	`require(taxAmount == burnAmount + liquidityAmount, "KAVIANL2::transfer: Burn value invalid");`
	
	Line 135
	`require(amount == sendAmount + taxAmount, "KAVIANL2::transfer: Tax value invalid");`

Description	Despite the risk of overflows being low in this contract, as the contract is using Solidity version 0.6.12, the use of raw addition may cause overflow issues in certain edge cases.
Recommendation	Consider either using Solidity versions 0.8.0 or higher, or implementing SafeMath's add rather than using raw addition.
Resolution	ACKNOWLEDGED

Issue #15	Incorrect comment
Severity	INFORMATIONAL

Location	Lines 23-24
	`// Max transfer amount rate in basis points. (default is 1% of total supply)`
	`uint16 public maxTransferAmountRate = 800;`

Description	The comment states that default is 1% of total supply, however it is actually 8%.
Recommendation	Consider correcting the comment.
Resolution	ACKNOWLEDGED

Issue #16	Excluding zero address can be removed
Severity	INFORMATIONAL
Description	Tokens cannot be sent to the zero address and the transaction would revert should there be an attempt to do so, as it is a common restriction in token contracts.
Recommendation	Consider removing this line as it is not required: `_excludedFromAntiwhale[address(0)] = true;`
Resolution	ACKNOWLEDGED

Issue #17	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 #18	`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 broken governance functionality issue, this can just be removed.
Resolution	ACKNOWLEDGED

Issue #19	Lack of events for `setExcludedFromAntiWhale`, `setLiquidityAddress` and `setSwapPair`
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

Issue #20	Functions can be made `external`
Severity	INFORMATIONAL
Description	`updateTransferTaxRate`, `updateBurnRate`, `updateMaxTransferAmountRate`, `updateMinAmountToLiquify`, `setExcludedFromAntiWhale`, `updateSwapAndLiquifyEnabled`, `setLiquidityAddress`, `setSwapPair` and `transferOperator` 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


PALADIN
BLOCKCHAIN SECURITY