PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Farmers Only (Onion Layer)

24 November 2021

paladinsec.co

info@paladinsec.co

Table of Contents

Table of Contents 2

Disclaimer 3

1 Overview 4

1.1 Summary 4

1.2 Contracts Assessed 4

1.3 Findings Summary 5

1.3.1 FarmersOnlyMasterChefL3 6

1.3.2 OnionCoin 6

2 Findings 7

2.1 FarmersOnlyMasterChefL3 7

2.1.1 Privileged Roles 9

2.1.2 Issues & Recommendations 10

2.2 OnionCoin 18

2.2.1 Token Overview 18

2.2.2 Privileged Roles 18

2.2.3 Issues & Recommendations 19

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 FarmersOnly's Onion layer farms on the Avalanche 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	Farmers Only (Onion layer)
URL	https://farmersonly.farm
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
MasterChef	0x7277046B34dE6c07A63057905C3a688712582BeB	✓ MATCH
OnionCoin	0xf6931e67EF0cB20423351A53069719Df2ad76e78	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🔴 High	1	1	-	-
🟠 Medium	4	3	1	-
🟡 Low	4	2	-	2
🟣 Informational	6	1	-	5
Total	**15**	**7**	**1**	**7**

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 FarmersOnlyMasterChefL3

ID	Severity	Summary	Status
01	HIGH	Users can avoid withdrawal fee through `emergencyWithdraw`	✓ RESOLVED
02	MEDIUM	Price oracle functionality to figure out the Onion supremacy can be manipulated at no cost	PARTIAL
03	MEDIUM	`rotateKing` could cause deposits and withdrawals to rotate if it is not called for six hours	✓ RESOLVED
04	MEDIUM	King pools could still be added after `isKingRotationActive` is set to false to potentially create more pools with a 20% withdrawal fee	✓ RESOLVED
05	MEDIUM	0x0 pools can still be added even after the setup is completed	✓ RESOLVED
06	LOW	`setAllocPoint`, `setKingPoolMul` and `disableKingRotation` might update rewards in hindsight	✓ RESOLVED
07	LOW	Unnecessary `onionusdt` initialization and `private` marking among other variables	ACKNOWLEDGED
08	LOW	King allocation points can still be adjusted manually by governance	✓ RESOLVED
09	INFO	Pool uses the contract balance to figure out the total deposits	ACKNOWLEDGED
10	INFO	Lack of events for `addStartPools`, `endNonNativesBoost`, `set`, `rotateKing`, `setHybridHarvest`, `disableKingRotation` and `setAllocPoint`	ACKNOWLEDGED
11	INFO	Usage of SafeMath on v0.8.0^ is unnecessary	ACKNOWLEDGED
12	INFO	Unnecessary `else` branch in `withdraw`	✓ RESOLVED
13	INFO	`pendingOnion` function does not account for maximum supply	ACKNOWLEDGED

1.3.2 OnionCoin

ID	Severity	Summary	Status
14	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	ACKNOWLEDGED
15	INFO	Symbols within the token name might not render correctly on Etherscan-compatible explorers	ACKNOWLEDGED

2 Findings

2.1 FarmersOnlyMasterChefL3

The FarmersOnly Layer 3 Masterchef is loosely based on the Panther Masterchef. It has some important changes.

Initially, there are 10 non-native pools with a deposit fee set to 4% (which cannot be set higher), then there are 5 pools with smaller deposit fees for tokens and LPs of their previous layers. Finally there are 6 pools which are marked as "king pools". Of these 6 pools, at any point in time, one pool will be marked as the "king". Within the current code, it is not yet clear which tokens will be put in these pools as the owner can initialize them once.

At any point in time, the designated king pool will have 100,000 allocation points while the 5 king candidate pools only have 100 allocation points — this significantly shifts the rewards to the king pool. While the king is active, a timer counts down from 6 hours (can be changed) whenever the total value locked in the Tomato and Corn staking pools exceeds the total value locked in the Onion staking pool. As long as Onion has superior TVL, the timer stays constant.

For the king pools, **a withdrawal fee of 20% applies** and users should take note when investing in these pools. No deposit-fee can be set. For the non-king pools, a deposit fee up to 4% can be set but no withdrawal fee can be set.

The governance can call the `endNonNativesBoost` function once to reduce the non-native pool weights. An initial emission rate of 0.0035 tokens per second applies and withdrawal fees are sent half to the fee address and half to the "locker address" which is outside the scope of this audit.

Finally, governance can enable and disable a special reward mechanism on pools. If it is disabled, the pools act like simple Panther pools which simple harvest lockups as is commonly known. If enabled, when the lockup is done, half of the unlocked tokens will be harvested and half will be relocked.

2.1.1 Privileged Roles

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

- `renounceOwnership`
- `transferOwnership`
- `addStartPools`
- `addPool`
- `setStartTime`
- `set`
- `setDevAddress`
- `setFeeAddress`
- `setLockAddress`
- `setVotesAddresses`
- `updateEmissionRate`
- `setKingRotInterval`

The following functions can be called by the first owner of the contract and will never be timelocked:

- `endNonNativesBoost [ callable once ]`
- `setOnionUsdt [ settable once ]`
- `setHybridHarvest`
- `setPoolLp [ settable once for every non initialized pool ]`
- `setKingPoolMul`
- `disableKingRotation [ callable once ]`
- `setAllocPoint`

2.1.2 Issues & Recommendations

Issue #01	Users can avoid withdrawal fee through emergencyWithdraw
Severity	● HIGH SEVERITY
Description	Presently, the emergencyWithdraw function does not account for the 20% withdrawal fee on king pools. This allows any user to avoid paying the withdrawal fee simply by calling emergencyWithdraw.
Recommendation	Consider also applying the withdrawal fee on emergencyWithdraw.
Resolution	✓ RESOLVED Withdrawal fee is now levied on emergencyWithdraw.

Issue #02	Price oracle functionality to figure out the Onion supremacy can be manipulated at no cost
Severity	● MEDIUM SEVERITY

Location	Lines 1634-1636
	`function price(ERC20 _token, address _pair, ERC20 _quote) private view returns (uint256) {`
	`    return ((_quote.balanceOf(_pair).mul(10**15)).div(_token.balanceOf(_pair)));`
	`}`

Description	Presently the function to calculate the USD price of the Onion, Corn and Tomato coins can be manipulated at pretty much zero cost. This is because the price function uses `balanceOf` to figure out the tokens within the Uniswap pair. `balanceOf` can be manipulated at no cost since tokens can be temporarily sent to a pair to be reclaimed completely for free by using `skim()`. Specifically, if a user were to want "Onion supremacy", they would write a simple contract that deposits USDT in the ONION/USDT pair, calls `update` to rotate the king and the king will automatically be flagged as safe as it sees that ONION is worth a lot of USDT. After the `updatePool` is called, they can simply reclaim all the USDT by calling `skim()` on the pair at zero cost. As this is all done within a single atomic transaction, nobody has an opportunity to reclaim the provided USDT. The impact of this issue is that the pricing functionality and all features linked to it can be freely manipulated. It should also be noted that the oracle relies on the USD token to always have 6 decimals, which will be the case in this deployment but might not be the case for forks or later layers. This should be considered when forking this contract for subsequent layers. The 1e15 precision might furthermore be insufficient in certain situations.
Recommendation	Consider using Uniswap's TWAP (or ChainLink) in the long term; in the short term, consider using the pair reserves instead of `balanceOf`, since this makes price manipulation slightly more expensive and will result in the issue being partially resolved.
Resolution	● PARTIALLY RESOLVED The client now uses `getReserves` and uses a slippage adjusted price.

Issue #03	`rotateKing` could cause deposits and withdrawals to rotate if it is not called for six hours
Severity	● MEDIUM SEVERITY
Location	Line 1651 `kingRotation = kingTimer - block.timestamp;`
Description	The above line can underflow if `block.timestamp` is greater than `kingTimer`. This causes the king rotation mechanism to revert which prevents `updatePool` from completing successfully. Users can still `emergencyWithdraw`.
Recommendation	Consider checking the case where `block.timestamp > kingTimer` before this statement.
Resolution	✓ RESOLVED

Issue #04	King pools could still be added after `isKingRotationActive` is set to false to potentially create more pools with a 20% withdrawal fee
Severity	● MEDIUM SEVERITY
Description	Presently, after `isKingRotation` active is set to `false`, more king pools can still be added even if they have a withdraw fee of 20%. This could be used in an attempt to steal some final money from users by creating more pools with 20% withdrawal fees and showing them on the frontend as if they are normal.
Recommendation	Consider adding more strict validation to the add function to no longer allow king pools to be added after the rotation is finished.
Resolution	✓ RESOLVED The client has increased their level of validation.

Issue #05	0x0 pools can still be added even after the setup is completed
Severity	● MEDIUM SEVERITY
Description	Presently, the contract does not prevent governance from adding more 0x0 pools, these are unique in the fact that they can be immediately initialized without going through timelock at a later time, which could be considered a governance risk as they could for example be initialized to a dummy token which only governance has. Furthermore, as long as there are 0x0 pools, `massUpdatePools` fails, preventing governance from making important calls to any functions that use it!
Recommendation	Consider only allowing 0x0 pool addition during contract creation. Furthermore, consider skipping updating 0x0 pools within `massUpdatePools`.
Resolution	✓ RESOLVED The whole 0x0 pool logic has been removed, simplifying the contract logic.

Issue #06	**`setAllocPoint`, `setKingPoolMul` and `disableKingRotation` might update rewards in hindsight**
Severity	● LOW SEVERITY
Description	Presently the `setAllocPoint` function does not call `updatePool` when it updates the reward weights of the pools. This could cause the new weight to apply in hindsight since the reward mechanism only updates the rewards whenever `updatePool` is called, for the time since the previous call to `updatePool`.
Recommendation	Consider calling `massUpdatePools` before all three functions. Using `updatePool` is insufficient since the all pools will be affected by the weight changes. `rotateKing` very importantly needs to update the old and new king pool as well before the change. Otherwise the new pool could receive rewards in hindsight, because `totalAllocPoint` should not change in this situation, `massUpdatePool` is not strictly required in this scenario (essentially their weights are just swapped).
Resolution	✓ RESOLVED `massUpdatePools` is used.

Issue #07	Unnecessary `onionusdt` initialization and private marking among other variables
Severity	● LOW SEVERITY
Location	Line 1399 `onionusdt = address(0);`
Description	The onionusdt pool is already set to zero initially, therefore there is no need to initialize it to zero explicitly. It is also marked as `private` among other variables, making them inaccessible for users to inspect.
Recommendation	Consider removing line 1399 and marking all private variables which do not have a public getter function as `public`.
Resolution	● ACKNOWLEDGED Multiple variables are still initialized in the contract.

Issue #08	King allocation points can still be adjusted manually by governance
Severity	● LOW SEVERITY
Description	The king allocation points are supposed to be 100,000. However, this can be adjusted manually using the governance functions.
Recommendation	Consider preventing governance from adjusting pool multipliers on the king pools. If there is a tokenomical reason that motivates such a need, this issue will also be marked as resolved.
Resolution	✓ RESOLVED The client has indicated that this is desired.

Issue #09	Pool uses the contract balance to figure out the total deposits
Severity	INFORMATIONAL
Description	As with pretty much all Masterchefs and staking contracts, the total number of tokens in the contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the masterchef.
Recommendation	Consider adding an `lpSupply` variable to the `PoolInfo` that keeps track of the total deposits.
Resolution	ACKNOWLEDGED

Issue #10	Lack of events for `addStartPools`, `endNonNativesBoost`, `set`, `rotateKing`, `setHybridHarvest`, `disableKingRotation` and `setAllocPoint`
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 #11	Usage of SafeMath on v0.8.0^ is unnecessary
Severity	INFORMATIONAL
Description	Using SafeMath on solidity versions higher or equal to v0.8.0 is unnecessary since the SafeMath functionality is now built in.
Recommendation	Consider using standard arithmetic.
Resolution	ACKNOWLEDGED

Issue #12	**Unnecessary else branch in withdraw**
Severity	● INFORMATIONAL

Location	Lines 1760-1762
	`} else {`
	`    withdrawFee = 0;`
	`}`

Description	Within the `withdraw` method, the `withdrawFee` is explicitly set to zero while it is already zero. This might waste some gas.
Recommendation	Consider simply setting it explicitly to zero during initialization. `uint256 withdrawFee = 0;`
Resolution	✓ RESOLVED

Issue #13	**`pendingOnion` function does not account for maximum supply**
Severity	● INFORMATIONAL
Description	The `pendingOnion` function currently does not account for the maximum supply mechanism. The UI must therefore incorporate the supply cap.
Recommendation	Consider adjusting the `pendingOnion` function with the exact same max supply logic as in `updatePool`.
Resolution	● ACKNOWLEDGED

2.2 OnionCoin

The OnionCoin is a simple ERC-20 token which allows for Onion tokens to be minted when the `mint` function is called by the owner of the contract, which at the time of deployment would be the deployer. Ownership is generally transferred to the Masterchef via the transferOwnership function, 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	0xf6931e67ef0cb20423351a53069719df2ad76e78
Token Supply	7,777 (enforced in Masterchef)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.2.2 Privileged Roles

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

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

2.2.3 Issues & Recommendations

Issue #14	**`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	● ACKNOWLEDGED This issue will be marked as resolved once token ownership has been transferred to the Masterchef.

Issue #15	**Symbols within the token name might not render correctly on Etherscan-compatible explorers**
Severity	● INFORMATIONAL
Description	To prevent phishing risk, Etherscan might not show non-alphadecimal characters correctly within the token name. The ' character within the token name FarmersOnly's Onion Coin might therefore be displayed weirdly on the explorer.
Recommendation	Consider whether this is an issue. If so, consider renaming the token.
Resolution	● ACKNOWLEDGED

PALADIN

BLOCKCHAIN SECURITY