PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Lydia Finance (IFO)

05 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 Ifo 6

2 Findings 7

2.1 Ifo 7

2.1.1 Issues & Recommendations 8

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 Lydia Finance's IFO contract 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	Lydia Finance (IFO)
URL	https://www.lydia.finance/launchpad
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`Ifo`	`Ifo.sol`	PENDING
`Source`	https://github.com/lydiafinance/lydia-ifo/blob/398917dffb615f929df17359dec7d9f4a0731927/contracts/Ifo.sol	

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1	-	-
Medium	2	1	1	-
Low	1	1	-	-
Informational	8	3	1	4
Total	**12**	**6**	**2**	**4**

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 Ifo

ID	Severity	Summary	Status
01	HIGH	Governance can withdraw all tokens as soon as the IFO ends causing users to potentially never get a chance to receive a refund or their allocated tokens	RESOLVED
02	MEDIUM	The UI function viewUserInfo is wrongly defined causing this function to potentially return wrong user-facing information	RESOLVED
03	MEDIUM	Unnecessary precision multipliers and rounding	PARTIAL
04	LOW	Contract does not work for deposits of reflective tokens	RESOLVED
05	INFO	viewUserOfferingAndRefundingAmountsForPools returns ambiguous array	ACKNOWLEDGED
06	INFO	Functions could revert due to division by zero exceptions in edge cases	ACKNOWLEDGED
07	INFO	Lack of constructor validation	ACKNOWLEDGED
08	INFO	Lack of indexing for PoolParametersSet pid event parameter	RESOLVED
09	INFO	msg.sender is unnecessarily cast to address(msg.sender)	RESOLVED
10	INFO	lpToken and offeringToken can be made immutable	RESOLVED
11	INFO	Misleading comments	PARTIAL
12	INFO	Lack of event for releaseTokens	RESOLVED

2 Findings

2.1 Ifo

The Lydia IFO contract can be used to execute IFOs on the Lydia platform. For each contract deployment, it allows Lydia to configure an `lpToken` which users can deposit into the IFO and an `offeringToken` which they will receive once its finished. Lydia can configure multiple pools for users to deposit in. Each pool can have an individual `lpToken` limit, `offeringToken` amount, per-wallet deposit limit and whether it has overflow taxes. Users can then choose to deposit in either of the two pools.

If a total amount greater than the raising amount is raised, this is called "overflow". Although it is permitted, it means that the offering amount will be divided over the depositors pro-rata their deposits and the deposited amount that exceeds the raising amount is refunded. However, a small (up to 1%) tax is levied on this overflow which gradually decreases as the overflow becomes larger. For example, the tax will only be 0.5% if the overflow is 50x the amount that was intended to be raised within that pool.

After the IFO has finished, the offering tokens are then released periodically by the IFO governance which means they are not all claimable at once, but the governance can unlock them in waves. It is up to the governance to decide at any point in time how much they want to release as there is no up-front schedule. Instead, there is simply a function which they can set to indicate the amount of unlocked tokens, this amount must be increasing so they cannot suddenly relock unlocked tokens. `lpTokens` deposited are sent to the governance while users receive the `offeringToken` in return.

2.1.1 Issues & Recommendations

Issue #01	**Governance can withdraw all tokens as soon as the IFO ends causing users to potentially never get a chance to receive a refund or their allocated tokens**
Severity	HIGH SEVERITY
Description	The governance function `finalWithdraw` can be called at any point of time. This function withdraws all provided LP tokens and offeringTokens which could result in the users being unable to withdraw their refund or allocated offering tokens. ❗ If the contract does not contain enough `offeringTokens` for the first allocated token withdrawal, the user can also never receive their refund.
Recommendation	Consider only allowing for `finalWithdraw` to be called once if `releasedPercent` is set to 100 and if `nextReleaseTimestamp` is at least 72 hours in the past to give everyone sufficient time to withdraw their refund (it might make sense to add some sanity checks to `nextReleaseTimestamp` as to not lock yourself out forever).
Resolution	RESOLVED During the first 48 hours after the IDO has finished, the governance can only withdraw the LP tokens that are not allocated for refunds, and they are also unable to withdraw any offering tokens. After these 48 hours have passed, the client still has full freedom over the contract balances so users should make sure to harvest within time.

Issue #02	The UI function viewUserInfo is wrongly defined causing this function to potentially return wrong user-facing information
Severity	MEDIUM SEVERITY

Location	Lines 421-426
	`for (uint8 i = 0; i < numberPools; i++) {`
	`  amountPools[i] = _userInfo[_user][i].amountPool;`
	`  statusPools[i] = _userInfo[_user][i].claimedPool;`
	`  purchasedPools[i] = _userInfo[_user][i].purchasedTokens;`
	`  claimedPools[i] = _userInfo[_user][i].claimedTokens;`
	`}`

Description	The `viewUserInfo` function does not use the `_pids` parameter to calculate the amount of iterations for the for loop, nor does it use `_pids[i]` to calculate the indices.
Recommendation	Consider using `_pids.length` instead of `numberOfPools`, consider using `[_pids[i]]` instead of `[i]` for the pid indices.
Resolution	RESOLVED The `[i]` occurrences have been replaced with `_pids[i]` and `numberPools` has been replaced with `_pids.length`.

Issue #03	Unnecessary precision multipliers and rounding
Severity	MEDIUM SEVERITY
Location	Line 557 `_userInfo[_user][_pid].amountPool.mul(1e18).div(_poolInformation[_pid].totalAmountPool.mul(1e6));`
Description	Line 557 unnecessarily does a multiplication and division while simply multiplying by 1e12 would result in the same precision. Furthermore, by using this rounded 1e12 precise percentage, severe rounding errors could occur if the two tokens have significantly different supplies. It might be better to avoid precision completely by using the mul-div pattern to calculate the amounts directly, this results in the maximum precision possible.
Recommendation	Consider using the mul-div pattern instead of a precision multiplier. The mul-div pattern reaches the maximum precision possible at the cost that it might possibly revert for very large numbers within the multiplication. Furthermore, consider calculating the refund amount in favour of the contract (currently it rounds up) to avoid any potential exploits where the user takes advantage of the rounding up of the refund. Otherwise consider removing the 1e18 and 1e6 multipliers in favor of a single 1e12 multiplier and being diligent in testing that this multiplier is sufficiently large for each individual IFO.
Resolution	PARTIALLY RESOLVED The client has fixed the typographical error with the 1e18 multiplication and 1e6 division but the precision remains low within the contract and is set to 1e12 throughout the contract. However, client has added test-cases to ensure that the rounding behavior is within bounds and will not cause issues. They will simply top up the contract with a few coins extra instead of implementing complex, error-prone logic everywhere. Paladin finds this reasonable.

Issue #04	Contract does not work for deposits of reflective tokens
Severity	LOW SEVERITY

Location	Lines 150-153
	`lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);`
	
	`// Update the user status`
	`_userInfo[msg.sender][_pid].amountPool = _userInfo[msg.sender][_pid].amountPool.add(_amount);`

Description	Currently the contract will add the `_amount` sent from the user to the contract to the user's deposited balance. However, with reflective tokens, this is not necessarily the amount received by the contract. In this instance, the user would be potentially eligible for a refund greater than the amount they deposited which could inadvertently cause them to steal other's deposits.

Recommendation	Consider using a before-after pattern to calculate the amount which was actually received by the contract. It should be noted that such a pattern should always be accompanied with reentrancy guards which in this case are already present.
	
	`uint256 before = lpToken.balanceOf(address(this));`
	`lpToken.safeTransferFrom(msg.sender, address(this), _amount);`
	`_amount = lpToken.balanceOf(address(this)).sub(before);`

Resolution	RESOLVED The recommended before-after pattern has been implemented.

Issue #05	`viewUserOfferingAndRefundingAmountsForPools` returns ambiguous array
Severity	INFORMATIONAL
Description	The `viewUserOfferingAndRefundingAmountsForPools` function returns an array of arrays where the subarrays have 3 indices. This might be confusing to third-party reviewers and users interacting through the explorer.
Recommendation	Consider returning an array of structs instead, to improve readability.
Resolution	ACKNOWLEDGED

Issue #06	Functions could revert due to division by zero exceptions in edge cases
Severity	INFORMATIONAL
Description	The code does not always correctly check if denominators within the divisions are non-zero, this could thus cause certain UI functions to revert while the pools are not yet initialized. The `_calculateTaxOverflow` function for example reverts if `_raisingAmountPool` is set to zero, and this causes the `viewPoolTaxRateOverflow` external function to revert while pools are not initialized. Furthermore, a `raisingAmountPool` of zero could revert `_calculateOfferingAndRefundingAmountsPool` which is used in the user facing `claimableTokens` function.
Recommendation	Consider adding non-zero checks for the aforementioned functions. Consider going through the codebase a second time to validate that all functions correctly work while certain variables might be zero. Consider adding test-cases for the different functions while the pools have not been initialized.
Resolution	ACKNOWLEDGED

Issue #07	Lack of constructor validation
Severity	INFORMATIONAL
Description	The constructor does not validate that `startTimestamp` is in the future and that `endTimestamp` is set after the `startTimestamp`.
Recommendation	Consider validating the above variables.
Resolution	ACKNOWLEDGED The client prefers to keep this portion of the contract as-is.

Issue #08	Lack of indexing for `PoolParametersSet` `pid` event parameter
Severity	INFORMATIONAL
Description	Essential identifying parameters within events should be marked as indexed. This allows for off-chain components to filter events only including these values.
Recommendation	Add indices to the key variables within the events you might want to filter on.
Resolution	RESOLVED

Issue #09	`msg.sender` is unnecessarily cast to `address(msg.sender)`
Severity	INFORMATIONAL
Description	`msg.sender` is cast to `address(msg.sender)` throughout the contract when used with `lpToken.safeTransfer()` and `lpToken.safeTransferFrom()`. This is unnecessary.
Recommendation	Consider replacing all occurrences of `address(msg.sender)` with `msg.sender`.
Resolution	RESOLVED The client has changed all occurrences to the shorter alternative.

Issue #10	`lpToken` and `offeringToken` 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 the above variables explicitly immutable.
Resolution	RESOLVED

Issue #11	Misleading comments
Severity	INFORMATIONAL

Description	Line 127
	`* @param _lpAmount: the number of LP token to withdraw (18 decimals)`
	
	The code contains a comment that indicates that the `lpAmount` must have 18 decimals, however, to our knowledge, there is no such enforcement nor is there a strict need for the LP token to have 18 decimals.
	
	Line 236 also contains a typographical error:
	`Release more tokens to allow IFO participants claim more`
	
	This should be "to claim more".

Recommendation	Consider removing the 18 decimals note from the comment. If you want to be clear to readers that they should not set this parameter to "1" for 1 token, consider indicating that this is the raw amount and not the decimal amount. Consider also fixing the typographical error.
Resolution	PARTIALLY RESOLVED The 18 decimals note has been updated but the client has incorrectly written "to claim more" next to the first location.

Issue #12	Lack of event for `releaseTokens`
Severity	INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications. ❗ `PoolParametersSet` also does not include `hasTax` as an event parameter.
Recommendation	Add an event for the above function.
Resolution	RESOLVED


PALADIN
BLOCKCHAIN SECURITY