PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For AvaLuan

20 November 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 AvaLuan, the next layer of the AvaTerra Farm, 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	AvaLuan
URL	https://luan.avaterra.finance/
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`LuanToken`	`0xe72c5547B2C66436D4Cb9Fd674001027A0d83A46`	MATCH
`MasterChef`	`0x9403B802dd7405d2ca4F501ec7A92c3a8b7051DE`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	1	1	-	
Informational	3	-	-	3
Total	**4**	**1**	**-**	**3**

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 LuanToken

ID	Severity	Summary	Status
01	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
02	INFO	UI function `pendingLuan` does not account for maximum supply cap	ACKNOWLEDGED
03	INFO	`LUANMAXSUPPLY` is not obtained from token contract	ACKNOWLEDGED
04	INFO	Gas optimization: updatePool calls token contract more than necessary	ACKNOWLEDGED

2 Findings

2.1 LuanToken

The LuanToken contract allows for LUAN tokens to be minted when the `mint` function is called by the contract Owner, who 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. In the constructor, 10 tokens are minted to the deployer for initial liquidity, and 10 tokens are minted to the treasury at 0x99Fb7c32020680AB5bea10061f4f1AD0e44d8690.

2.1.1 Token Overview

Address	0xe72c5547B2C66436D4Cb9Fd674001027A0d83A46
Token Supply	3,000 (enforced in Masterchef)
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None
Pre-mints	20

2.1.2 Privileges

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

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

2.1.3 Issues & Recommendations

Issue #01	**`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 is to be 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 20 tokens were pre-minted and ownership of the token has been transferred to the Masterchef.

2.2 MasterChef

The AvaLuan Masterchef is a modified fork of Goose Finance's Masterchef. Deposit fees have been limited to a maximum of 4%. There is a maximum supply cap of LUAN tokens which can be minted, enforced at the Masterchef level, of 3000 tokens.

2.2.1 Privileges

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

- `add`
- `set`
- `updateEmissionRate`
- `updateStartBlock`
- `updateTreasuryAddress`
- `updateFeeAddress`
- `updateStartTime`
- `transferOwnership`
- `renounceOwnership`

2.2.2 Issues & Recommendations

Issue #02	**UI function `pendingLuan` does not account for maximum supply cap**
Severity	INFORMATIONAL
Description	The UI function `pendingLuan` does not account for the fact that minting stops at a certain point. This might confuse people shortly after the minting has stopped as the UI will still show them some rewards for a short period, if the frontend does not have additional logic. As `getMultiplier` already adjusts to zero, this issue is especially minimal as the only scenario that is affected is if the supply is nearly reached and less would be minted.
Recommendation	Consider incorporating similar logic to the `updatePool` logic in `pendingLuan`.
Resolution	ACKNOWLEDGED

Issue #03	**LUANMAXSUPPLY is not obtained from token contract**
Severity	INFORMATIONAL

Location	Line 913
	`uint256 private constant LUANMAXSUPPLY = 3000 ether;`
	
	Line 1040~
	`if (LuanCalc <= LUANMAXSUPPLY) {`
	`  luan.mint(treasuryAdd, toTreasury);`
	`  luan.mint(address(this), luanReward);`
	`} else {`
	`  luanReward = LUANMAXSUPPLY - luan.totalSupply();`
	`  luan.mint(address(this), luanReward);`
	`}`

Description	`LUANMAXSUPPLY` is hardcoded as a constant in the Masterchef contract instead of getting the value from the token's `MAXSUPPLY` value. Although in this deployment, the token contract and Masterchef contract both have the same value, it is recommended for the variable used in the Masterchef contract to obtain the value from the token contract.
Recommendation	`LUANMAXSUPPLY` can be made immutable and set in the constructor from the value returned by `maxSupply()` in the token contract. `LUANMAXSUPPLY  = _luan.maxSupply();`
Resolution	ACKNOWLEDGED

Issue #04	**Gas optimization: `updatePool` calls token contract more than necessary**
Severity	INFORMATIONAL

Location	(Example) Line 1038~
	`uint256 LuanCalc = luan.totalSupply() + luanReward + toTreasury;`
	
	`if (LuanCalc <= LUANMAXSUPPLY) {`
	`luan.mint(treasuryAdd, toTreasury);`
	`luan.mint(address(this), luanReward);`
	`} else {`
	`luanReward = LUANMAXSUPPLY - luan.totalSupply();`
	`luan.mint(address(this), luanReward);`
	`}`

Description	Within `updatePool`, the `totalSupply` function is called multiple times on the token, this wastes unnecessary gas.
Recommendation	Consider storing the `totalSupply` variables in local variables that can be reused throughout the contract.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY