



PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For The Dragon's Lair

02 September 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 Masterchef	6
1.3.2 DragonEggToken	6
1.3.3 Timelock	6
2 Findings	7
2.1 MasterChef	7
2.1.1 Privileged Roles	7
2.1.2 Issues & Recommendations	8
2.2 DragonEggToken	10
2.2.1 Token Overview	10
2.2.2 Privileged Roles	10
2.2.3 Issues & Recommendations	10
2.3 Timelock	11
2.3.1 Issues & Recommendations	11



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 advocacy 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 Dragon's Lair 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	The Dragon's Lair
URL	https://thedragonslair.farm/
Platform	Avalanche
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
MasterChef	0x...846Db264	✓ MATCH
DragonEggToken	0x...DD079c57	✓ MATCH
Timelock	0x...B85C4Ba5	✓ MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	0	-	-	-
● Medium	0	-	-	-
● Low	1	1	-	-
● Informational	2	2	-	-
Total	3	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 Masterchef

ID	Severity	Summary	Status
01	LOW	The pendingDragonEgg function will revert if totalAllocPoint is zero	RESOLVED
02	INFO	dragonEgg can be made immutable	RESOLVED
03	INFO	Minting does not account for feeAddress allocation	RESOLVED

1.3.2 DragonEggToken

No issues found.

1.3.3 Timelock

No issues found.



2 Findings

2.1 MasterChef

The Dragon's Lair Masterchef contract was forked from PolyWantsACracker, which was previously audited by Paladin. As such, it is a secure Masterchef contract and we commend Dragon's Lair on forking an audited, proven Masterchef. Deposit fees have an upper limit of 3.01%, transfer tax tokens are properly accounted for, and the migrator function has also been removed.

A notable feature of this Masterchef is that the maximum token supply of 250,000 tokens is enforced in the `updatePool` function, which halts minting should the maximum supply be reached. Additionally, the use of Solidity version `^0.8.0` means that [overflow checks are built-in](#).



2.1.1 Privileged Roles

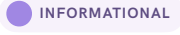

The following functions can be called by the owner of the contract:

- `add`
- `set`
- `setStartBlock`
- `setFeeAddress`



2.1.2 Issues & Recommendations

Issue #01	The pendingDragonEgg function will revert if totalAllocPoint is zero
Severity	 LOW SEVERITY
Description	In the pendingDragonEgg 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: <pre>if (block.number > pool.lastRewardBlock && lpSupply != 0 && totalAllocPoint > 0) {</pre>
Resolution	 RESOLVED

Issue #02	dragonEgg 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 these variables explicitly immutable.
Resolution	 RESOLVED

Issue #03**Minting does not account for feeAddress allocation****Severity** INFORMATIONAL**Location**Lines 189-190

```
else if ((dragonEgg.totalSupply() + dragonEggReward) >
dragonEggMaximumSupply)
    dragonEggReward = dragonEggMaximumSupply -
dragonEgg.totalSupply();
```

Description

The minting of the native DragonEgg tokens is done in the updatePool function. There is an upper limit of 250,000 tokens, which if reached, will stop rewards minting in the Masterchef. Unfortunately, as one-tenths of tokens are minted to the feeAddress, this may result in dragonEgg.totalSupply slightly exceeding dragonEggMaximumSupply because the current accounting does not factor in 110% of token minting (100% to the Masterchef address, and 10% to the feeAddress).

Recommendation

The simplest method for ensuring that the token supply does not exceed the maximum supply is by altering the minting allocations such that the Masterchef mints 10% to the feeAddress and the remaining 90% to stakers, and thus the sum is 100% :

```
if (dragonEggReward > 0)
{
    dragonEgg.mint(feeAddress, dragonEggReward / 10);
    dragonEgg.mint(address(this), dragonEggReward * 9/10);
}
```

Resolution RESOLVED

The client has revamped the updatePool function to now correctly implement 10% minting to the dev and 100% to stakers.

2.2 DragonEggToken

The DragonEggToken is a simple ERC-20 token.

2.2.1 Token Overview

Address	0x...DD079c57
Token Supply	250,000 (two hundred and fifty thousand)
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None
Pre-mints	1000 tokens preminted to 0x306e5F7FAe63a86b3E2D88F94cCa8D7614684D91

2.2.2 Privileged Roles

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

- mint

2.2.3 Issues & Recommendations

No issues found.

2.3 Timelock

The Timelock contract is a clean fork of Compound Finance's timelock. This is the most common contract used in DeFi to time lock governance access and is thus compatible with most third-party tools.

Parameter	Value	Description
Delay	TBC	The delay indicates the time the administrator has to wait after queuing a transaction to execute it.
Minimum Delay	6 hours	<p>The minDelay indicates the lowest value that the delay can minimally be set.</p> <p>Sometimes, projects will queue a transaction that sets the delay to zero with the hope that nobody notices it. However, because of the minimum delay parameter, the value of delay can never be lower than that of the minDelay value. Note that the administrator could still queue a transaction to simply transfer the ownership back to their own account so it is still important to inspect every transaction carefully.</p>
Maximum Delay	30 days	The maximum delay indicates the highest value that the delay can be set.
Grace Period	14 days	After the delay has expired after queuing a transaction, the administrator can only execute it within the grace period. This is to prevent them from hiding a malicious transaction among much earlier transactions, hoping that it goes unnoticed or buried, which can be executed in the future.

2.3.1 Issues & Recommendations

No issues found.



PALADIN
BLOCKCHAIN SECURITY