

PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For PolyPup Collar

03 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 CollarToken 6

1.3.2 MasterChef 6

2 Findings 7

2.1 CollarToken 7

2.1.1 Token Overview 7

2.1.2 Privileged Roles 8

2.1.3 Issues & Recommendations 9

2.2 MasterChef 12

2.2.1 Privileged Roles 12

2.2.2 Issues & Recommendations 13

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 PolyPup's Collar layer. 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.

At the time of assessment, the contracts were sent to Paladin via a GitHub repository and may differ from the ones deployed on the blockchain.

1.1 Summary

Project Name	PolyPup Collar
URL	https://polypup.finance/
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`CollarToken`	`0x8df26a1bd9bd98e2ec506fc9d8009954716a05dc`	MATCH
`MasterChef`	`0xdabdf071635df85e87cb5b531c4db448a0f0ce21`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	3	3	-	-
Informational	6	3	-	3
Total	**9**	**6**	**0**	**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 CollarToken

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
02	INFO	Governance functionality is broken	ACKNOWLEDGED
03	INFO	`delegateBySig` can be frontrun and cause denial of service (present in all Goose forks)	ACKNOWLEDGED

1.3.2 MasterChef

ID	Severity	Summary	Status
04	LOW	Setting `devAddress` to the zero address will break the `deposit` and `withdraw` functions	RESOLVED
05	LOW	`updateEmissionRate` has no maximum safeguard	RESOLVED
06	INFO	`collar` can be made `immutable`	ACKNOWLEDGED
07	INFO	`BONUS_MULTIPLIER` looks to be redundant	RESOLVED
08	INFO	Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions	RESOLVED
09	INFO	Contract uses raw subtraction	RESOLVED

2 Findings

2.1 CollarToken

The contract allows for COLLAR tokens to be minted when the `mint` function is called by the Owner, who at the time of deployment would be the deployer. However, 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.

2.1.1 Token Overview

Address	0x8df26a1bd9bd98e2ec506fc9d8009954716a05dc
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None

2.1.2 Privileged Roles

The owner of the CollarToken contract should be transferred to the Masterchef.

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

- `mint`

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 10,000 tokens have been pre-minted and token ownership has been transferred to the Masterchef.

Issue #02	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 mistake is present in pretty much every single farm out there including PancakeSwap and even sushiswap.
Recommendation(s)	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 #03	**delegateBySig can be frontrun and cause denial of service (present in all Goose forks)**
Severity	INFORMATIONAL
Location	Line 118 `require(nonce == nonces[signatory]++, "EGG::delegateBySig: invalid nonce");`
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, consider removing this section of the contract as snapshot.org is a more viable alternative.
Resolution	ACKNOWLEDGED

2.2 MasterChef

The Masterchef is a fork of Goose Finance's Masterchef. A notable feature of forking this Masterchef is the removal of the migrator function from Pancakeswap, which of late has been used maliciously to steal users' tokens. We commend Polypup Collar on their decision to fork a relatively safer version of the Masterchef. Finally, the deposit fees have an upper limit of 4%, which is a reasonable cap.

2.2.1 Privileged Roles

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

- `add`
- `set`
- `setDevAddress`
- `setFeeAddress`
- `updateEmissionRate`
- `updateStartBlock`

2.2.2 Issues & Recommendations

Issue #04	**Setting `devAddr` to the zero address will break the `deposit` and `withdraw` functions**
Severity	LOW SEVERITY
Description	Any attempt to transfer or mint tokens to the zero address will revert, thus causing deposits and withdrawals to revert if the `devAddr` is ever set to the zero address.
Recommendation(s)	To prevent this from ever happening by accident and to limit governance risks, consider adding a requirement like the following: `require(_devddr != address(0), "!nonzero");` to the `setDevAddress` function.
Resolution	RESOLVED

Issue #05	**`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(s)	Consider adding a `MAX_EMISSION_RATE` variable and setting it to a reasonable value. `require(_CollarPerBlock <= MAX_EMISSION_RATE,"Too high");`
Resolution	RESOLVED There is now an upper limit of 2 tokens per block.

Issue #06	`collar` 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(s)	Consider making collar explicitly `immutable`.
Resolution	ACKNOWLEDGED

Issue #07	`BONUS_MULTIPLIER` looks to be 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(s)	Consider removing `BONUS_MULTIPLIER` and associated comments.
Resolution	RESOLVED

Issue #08	Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions
Severity	INFORMATIONAL
Description	Within `updatePool`, `accCollarPerShare` is based on the `lpSupply` variable. `pool.accCollarPerShare = pool.accCollarPerShare.add(collarReward.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(s)	Consider increasing precision to 1e18 across the entire contract.
Resolution	RESOLVED

Issue #09	Contract uses raw subtraction
Severity	INFORMATIONAL
Location	Line 1722 `_amount = pool.lpToken.balanceOf(address(this)) - balanceBefore;`
Description	Although the risk of underflow is low, it is considered best practice to use SafeMath rather than raw addition and subtraction in arithmetic operations.
Recommendation(s)	Consider using SafeMath's sub rather than raw subtraction.
Resolution	RESOLVED


PALADIN
BLOCKCHAIN SECURITY