PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Preliminary Report

For PolyGamma Finance

20 October 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 PolyGamma Finance on the Polygon 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	PolyGamma Finance
URL	https://polygamma.finance/
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`GammaToken`	`0x329f5e8aff351327e63acdb264389c798a46c2d3`	MATCH
`MasterChef`	`0x662ae0f1ba0e2da3bd8f664b579055c84b8a57a5`	MATCH

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	0	-	-	-
Medium	0	-	-	-
Low	1	1	-	-
Informational	3	2	-	1
Total	**4**	**3**	**-**	**1**

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 GammaToken

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	`_totalSupply` is sufficient to keep track of the total token supply minted	RESOLVED

1.3.2 MasterChef

ID	Severity	Summary	Status
03	INFO	Total token supply might not be minted due to try and catch pattern	ACKNOWLEDGED
04	INFO	`MAX_EMISSION_RATE` can be declared as a constant	RESOLVED

2 Findings

2.1 GammaToken

The GammaToken is a simple ERC20 token.

2.1.1 Token Overview

Address	0x329f5e8aff351327e63acdb264389c798a46c2d3
Token Supply	70,000
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None
Pre-mints	3,500

2.1.2 Privileged Roles

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 tokens for dumping.
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	RESOLVED 3,500 tokens were pre-minted and ownership of the contract has been transferred to the Masterchef.

Issue #02	**`_totalSupply` is sufficient to keep track of the total token supply minted**
Severity	INFORMATIONAL
Description	Both `MAXCAP` and `_totalSupply` are incremented by the amount of tokens minted, and decremented by the amount of tokens burnt. This makes `MAXCAP` redundant.
Recommendation	Consider removing `MAXCAP`, and keeping track of the token supply using `_totalSupply`. If the emissions should stop when the `MAXCAP` is minted (eg. on burns new emissions should not be possible again), consider removing the `MAXCAP` reduction on burn.
Resolution	RESOLVED `MAXCAP` has been removed.

2.2 MasterChef

The MasterChef contract was forked from PolyBeta, which was previously audited by Paladin. As such, it is a battle-tested and secure Masterchef. Most notably, there are no hard rug risk functionalities within the contract. Deposit fees have an upper limit of 4%, and the upper limit of 11,400 tokens is enforced via the try/catch implementation in the `updatePool` function.

2.2.1 Privileged Roles

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

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

2.2.2 Issues & Recommendations

Issue #03	Total token supply might not be minted due to try and catch pattern
Severity	INFORMATIONAL

Description	As there is a MAXCAPSUPPLY for the Gamma token, minting the reward and causing the max cap to exceed would result in a revert.
	
	`require(MAXCAP.add(amount) <= MAXCAPSUPPLY, "Max supply reached");`
	
	To prevent this, the following try-catch pattern is done in `updatePool`.
	
	`try gamma.mint(devaddr, gammaReward.div(10)) { } catch (bytes memory reason) { gammaReward = 0; emit GammaMintError(reason); }`
	
	`try gamma.mint(address(this), gammaReward) { } catch (bytes memory reason) { gammaReward = 0; emit GammaMintError(reason); }`
	
	In the case where `MAXCAP + amount` does exceed `MAXCAPSUPPLY`, the mint will not be done. This means that the token supply could be capped at an amount slightly lower than `MAXCAPSUPPLY`.

Recommendation	Consider minting the difference between `MAXCAPSUPPLY` and `MAXCAP`, if any.
Resolution	ACKNOWLEDGED

Issue #04	`MAX_EMISSION_RATE` can be declared as a `constant`
Severity	INFORMATIONAL
Description	As `MAX_EMISSION_RATE` is only declared once as a state variable and never changed, it can be declared as a constant for gas optimization.
Recommendation	Add the constant keyword when declaring `MAX_EMISSION_RATE` .
Resolution	RESOLVED `MAX_EMISSION_RATE` has been declared constant.


PALADIN
BLOCKCHAIN SECURITY