



PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Preliminary Report

For Palladium Farm

18 October 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 Palladium	6
1.3.2 MasterChef	6
2 Findings	7
2.1 Palladium	7
2.1.1 Token Overview	7
2.1.2 Privileged Roles	7
2.1.3 Issues & Recommendations	9
2.2 MasterChef	11
2.2.1 Privileged Roles	11
2.2.2 Issues & Recommendations	12



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 Palladium Farm on the Binance Smart Chain (BSC). 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	Palladium Farm
URL	https://palladium.farm/
Platform	Binance Smart Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
Palladium	Palladium.sol	PENDING
MasterChef	MasterChef.sol	PENDING
Source	https://github.com/palladiumfarm/Palladium-Farm	

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	0	-	-	-
● Medium	1	1	-	-
● Low	2	1	-	1
● Informational	5	1	-	4
Total	8	3	-	5

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 Palladium

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	ACKNOWLEDGED
02	INFO	maxTransferAmountRate is not used	ACKNOWLEDGED
03	INFO	_moveDelegates is not done on token transfers and burns	ACKNOWLEDGED

1.3.2 MasterChef

ID	Severity	Summary	Status
04	MEDIUM	set allows modifying to a value beyond MAXIMUM_HARVEST_INTERVAL	RESOLVED
05	LOW	updateEmissionRate has no maximum safeguard	RESOLVED
06	INFO	emergencyWithdraw does not adhere to the checks effects interactions pattern	RESOLVED
07	INFO	Pool uses the contract balance to figure out the total deposits	ACKNOWLEDGED
08	INFO	Total token supply could be over minted	ACKNOWLEDGED

2 Findings

2.1 Palladium

The Palladium token is a simple ERC-20 token which will be used as the main reward token for the Masterchef. Palladium tokens will be minted when the `mint` function is called by the owner of the contract, which at the time of deployment would be the Palladium team. Users should therefore carefully inspect that this account has been correctly initialized to the Masterchef. The token has a maximum supply of 200,000.

2.1.1 Token Overview

Address	TBD
Token Supply	200,000
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None

2.1.2 Privileged Roles

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

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

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


- `updateMaxTransferAmountRate`
- `setExcludedFromAntiWhale`
- `transferOperator`



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

 ACKNOWLEDGED

This issue would be marked as resolved once the owner of the contract has been transferred to the Masterchef.

Issue #02 **maxTransferAmountRate is not used**

Severity ● LOW SEVERITY

Description Although `_maxTransferAmountRate` is set, it is never used in the `antiWhale` modifier. Instead, `maxTransferAmount` is used, which always returns the total supply.

Recommendation Consider removing the `antiWhale` functionality since it is not functional currently.

 If the `antiWhale` functionality is to be used, `maxTransferAmountRate` should be used to calculate the maximum amount of tokens per transfer. There should be a reasonable lower limit for `maxTransferAmountRate`.

Resolution ● ACKNOWLEDGED

Issue #03 **_moveDelegates is not done on token transfers and burns**

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 issue is present in pretty much every single farm out there including PancakeSwap and even SushiSwap.

Recommendation 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

2.2 MasterChef

The Palladium masterchef is a modified Panther masterchef fork. Just like Panther, rewards can only be harvested after an interval, which is configurable to at most 1 day. There is a maximum of 4% deposit fees. There is also a referral bonus of up to 4%



There is also a maximum supply cap of Palladium tokens which can be minted, enforced at the masterchef level, of 2,000,000 tokens.

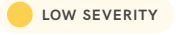

2.2.1 Privileged Roles

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

- add
- set
- setDevAddress
- setFeeAddress
- updateEmissionRate
- updateStartBlock
- transferOwnership
- renounceOwnership

2.2.2 Issues & Recommendations

Issue #04	set allows modifying harvestInterval to a value beyond MAXIMUM_HARVEST_INTERVAL
Severity	 MEDIUM SEVERITY
Description	<p>Although add has the following check to ensure that the harvestInterval set is lesser than or equals to MAXIMUM_HARVEST_INTERVAL, this is lacking in the set function:</p> <pre>require(_harvestInterval <= MAXIMUM_HARVEST_INTERVAL, "add: invalid harvest interval");</pre> <p>The owner is thus able to change the harvestInterval of an existing pool to an extremely high value, preventing users from harvesting.</p>
Recommendation	Add the same check in the set function to ensure that the harvestInterval will not exceed MAXIMUM_HARVEST_INTERVAL.
Resolution	 RESOLVED The same check has been added in the set function.

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	Consider adding a MAX_EMISSION_RATE variable and setting it to a reasonable value. <pre>require(_PalladiumPerBlock <= MAX_EMISSION_RATE, "Too high");</pre>
Resolution	 RESOLVED A MAX_EMISSION_RATE of 1 token per block check has been added.

Issue #06**emergencyWithdraw does not adhere to the checks effects interactions pattern****Severity** INFORMATIONAL**Description**

emergencyWithdraw does the external safeTransfer call before setting all the user's state data to 0. As there is a reentrancy guard to prevent reentrancy, this is marked as informational. However, it is recommended to follow the checks-effects-interactions pattern.

Recommendation

Consider making the following change:

```
uint256 amount = user.amount;
user.amount = 0;
user.rewardDebt = 0;
user.rewardLockedUp = 0;
user.nextHarvestUntil = 0;
pool.lpToken.safeTransfer(address(msg.sender), amount);
emit EmergencyWithdraw(msg.sender, _pid, amount);
```

Resolution RESOLVED


A change has been made to adhere to the checks effects interactions pattern in the emergencyWithdraw function.

Issue #07**Pool uses the contract balance to figure out the total deposits****Severity** INFORMATIONAL**Description**

As with pretty much all Masterchefs and staking contracts, the total number of tokens in the contract is used to determine the total number of deposits. This can cause dilution of rewards when people accidentally send tokens to the masterchef.

Recommendation

Consider adding an lpSupply variable to the PoolInfo that keeps track of the total deposits.

Resolution ACKNOWLEDGED

Issue #08**Total token supply could be over minted****Severity** INFORMATIONAL**Description**

The following check is done in `getMultiplier`, returning 0 if the sum of `totalSupply` and `totalLockedUpRewards` is greater than or equals to `MAX_SUPPLY_CAP`.

```
if (IERC20(palladium).totalSupply() + totalLockedUpRewards
    >= MAX_SUPPLY_CAP) {
    return 0;
}
```

However, this does not account for `ReferralCommission` minted, which does not rely on the value returned by `getMultiplier`.

Also, in the unlikely case where the sum of `totalSupply` and `totalLockedUpRewards` results in an integer overflow, it would be possible to over mint.

Recommendation

Consider adding the supply check before each mint is done, even for the referral commission.

Also, for the addition of `totalSupply` and `totalLockUpRewards`, it is recommended to use `SafeMath`.

Resolution ACKNOWLEDGED



PALADIN
BLOCKCHAIN SECURITY