PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For AstroFarms Finance (Virgo)

14 August 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 the Virgo layer of AstroFarms Finance. 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	AstroFarms - Virgo layer
URL	https://astrofarms.finance
Platform	Polygon
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
NeptuneToken	0x4527d831cefc76d0e5f8699f8ff4494611a6bf31	✓ MATCH
LeoToken	0xbE8DAb8Ce8521ecFDe43a8Ff8d5C6644F4dCECb7	✓ MATCH
CancerToken	0x056bdcd1d8436ae303023eade224f91825bf8e43	✓ MATCH
VirgoToken	VirgoToken.sol	
MasterChef	VirgoMasterChefTest.sol	

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	1	1	-	-
● Medium	1	1	-	-
● Low	6	4	-	2
● Informational	6	3	-	3
Total	**14**	**9**	**0**	**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 NeptuneToken

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	INFORMATIONAL	Governance functionality is broken (present in all Goose forks)	ACKNOWLEDGED

1.3.2 LeoToken

ID	Severity	Summary	Status
03	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	ACKNOWLEDGED
04	INFORMATIONAL	Governance functionality is broken (present in all Goose forks)	ACKNOWLEDGED

1.3.3 CancerToken

ID	Severity	Summary	Status
05	LOW	`mint` function can be used to pre-mint large amounts of tokens before ownership is transferred to the Masterchef	✓ RESOLVED
06	INFORMATIONAL	Governance functionality is broken (present in all Goose forks)	ACKNOWLEDGED

1.3.4 VirgoToken

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

1.3.4 MasterChef

ID	Severity	Summary	Status
08	HIGH	Malfunctioning bonus logic design	✓ RESOLVED
09	MEDIUM	No underflow protection in `deposit` before-after calculation	✓ RESOLVED
10	LOW	Setting `devAddress` to the zero address will break `updatePool`	✓ RESOLVED
11	LOW	updateEmissionRate has no maximum safeguard	✓ RESOLVED
12	INFORMATIONAL	Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions	✓ RESOLVED
13	INFORMATIONAL	`virgo`, `forgeTokenLeo`, `nftIdLeo`, `forgeTokenCancer`, `nftIdCancer`, `forgeTokenNeptune` and `nftIdNeptune` can be made `immutable`	✓ RESOLVED
14	INFORMATIONAL	Outdated comment above `BONUS_MULTIPLIER`	✓ RESOLVED

2 Findings

2.1 NeptuneToken

2.1.1 Token Overview

Address	0x4527d831cefc76d0e5f8699f8ff4494611a6bf31
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.1.2 Privileged Roles

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

- `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 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 The client has indicated that these tokens have been used for their previous farms and ownership has been transferred a long time ago.

Issue #02	Governance functionality is broken (present in all Goose forks)
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, which is used by many of the larger projects.
Resolution	● ACKNOWLEDGED

2.2 LeoToken

2.2.1 Token Overview

Address	0xbE8DAb8Ce8521ecFDe43a8Ff8d5C6644F4dCECb7
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.2.2 Privileged Roles

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

- `mint`

2.2.3 Issues & Recommendations

Issue #03	**`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 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	● ACKNOWLEDGED

Issue #04	Governance functionality is broken (present in all Goose forks)
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, which is used by many of the larger projects.
Resolution	● ACKNOWLEDGED

2.3 CancerToken

2.3.1 Token Overview

Address	0x056bdcd1d8436ae303023eade224f91825bf8e43
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.3.2 Privileged Roles

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

- `mint`

2.3.3 Issues & Recommendations

Issue #05	**`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 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 The client has indicated that these tokens have been used for their previous farms and ownership has been transferred a long time ago.

Issue #06	Governance functionality is broken (present in all Goose forks)
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, which is used by many of the larger projects.
Resolution	● ACKNOWLEDGED

2.4 VirgoToken

2.4.1 Token Overview

Address	Not deployed
Token Supply	Unlimited
Decimal Places	18
Transfer Max Size	None
Transfer Min Size	None
Transfer Fees	None

2.4.2 Privileged Roles

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

- `mint`

2.4.3 Issues & Recommendations

Issue #07	**`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 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	● ACKNOWLEDGED

2.5 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. Furthermore, in comparison to Goose Finance, AstroFarms has limited the deposit fee to at most 4%. We commend AstroFarms on their decision to fork a relatively safer version of the Masterchef and trim down the governance privileges with regards to the deposit fees. AstroFarms has also incoporated many of our best practices before commissioning this audit report.

The main difference of AstroFarm's Masterchef from other traditional Goose-based Masterchefs is that if any of the three NFT tokens are held during harvesting, a reward bonus will be awarded:

- Leo VIP: 50% harvest bonus
- Cancer VIP: 25% harvest bonus
- Neptune VIP: 25% harvest bonus

2.5.1 Privileged Roles

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

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

2.5.2 Issues & Recommendations

Issue #08	Malfunctioning bonus logic design
Severity	● HIGH SEVERITY

Location	Lines 355-360, 386-391
	`user.rewardDebt = user`
	`    .amount`
	`    .mul(pool.accVirgoPerShare)`
	`    .div(1e12)`
	`    .mul(user.bonusMultiplier)`
	`    .div(100);`

Description	Currently for the user, their `bonusMultiplier` is added to the `rewardDebt` variable, making their `rewardDebt` greater if they hold an NFT. However, the pools' reward balance is based on the assumption that the base rate is also scaled appropriately. Since at the end of a deposit or withdrawal, the `user.rewardDebt` should match the scaled `pool.accVirgoPerShare`, this extra multiplication causes a violation of the core business logic.

Recommendation	Consider removing all usage of the NFT reward mechanism first to refactor it. Instead of trying to incorporate it in the pending and `rewardDebt` variables directly, consider incorporating it in the following manner:
	
	Lines 333-335, line 378-380
	`if (pending > 0) {`
	`    uint256 multiplier = calculateBonus(msg.sender);`
	`    if (multiplier != 100) {`
	`        uint256 bonus =`
	`pending.mul(multiplier).div(100).sub(pending);`
	`         virgo.mint(address(this), bonus);`
	`        pending = pending.add(bonus);`
	`    }`
	`    safeVirgoTransfer(msg.sender, pending);`
	`}`
	
	Note that the `user.bonusMultiplier` variable can be removed as well since it does not serve any specific purpose.

Resolution	✓ RESOLVED The recommended code block has been implemented.

Issue #09	No underflow protection in `deposit` before-after calculation
Severity	● LOW SEVERITY
Location	Line 344 `_amount = pool.lpToken.balanceOf(address(this)) - balanceBefore;`
Description	Within the `deposit` function, a before-after pattern is used to calculate the deposited amount. Even though this is good idea, the before-after calculation can underflow in the very unlikely case that there are less tokens after then before the deposit. Although this case is extremely unlikely, this underflow possibility would violate one of the core principles of the Masterchef that makes it safe to use for users (the fact that the amount calculations are trivial and predictable) and should thus be taken out.

Recommendation	Consider using SafeMath instead of raw subtraction.
	Line 344
	`_amount =`
	`pool.lpToken.balanceOf(address(this)).sub(balanceBefore);`

Resolution	✓ RESOLVED SafeMath is now used.

Issue #10	Setting `devAddress` to the zero address will break `updatePool`
Severity	● LOW SEVERITY
Description	Minting tokens to the zero address will revert transactions. Most deposits and withdrawals will thus revert if the `devAddr` is ever set to the zero address.
Recommendation	To prevent this from ever happening by accident and to limit governance risks, consider adding a requirement like the following : `require(_devAddress != address(0), "!nonzero");` to the `setDevAddress` function.
Resolution	✓ RESOLVED

Issue #11	`updateEmissionRate` has no maximum safeguard
Severity	● LOW SEVERITY

Location	Lines 440-444
	`function updateEmissionRate(uint256 _VirgoPerBlock) external onlyOwner {`
	`    massUpdatePools();`
	`    VirgoPerBlock = _VirgoPerBlock;`
	
	`    emit UpdateEmissionRate(msg.sender, _VirgoPerBlock);`
	`}`

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. `require(_virgoPerBlock <= MAX_EMISSION_RATE,"Too high");`
Resolution	✓ RESOLVED The emission rate can now be updated to at most 1 token per block.

Issue #12	**Rounding vulnerability to tokens with a very large supply can cause large supply tokens to receive zero emissions**
Severity	● INFORMATIONAL
Description	Within `updatePool`, `accVirgoPerShare` is based on the `lpSupply` variable. `pool.accVirgoPerShare = pool.accVirgoPerShare.add(virgoReward.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	Consider increasing precision to 1e18 across the entire contract.
Resolution	✓ RESOLVED

Issue #13	**`virgo`, `forgeTokenLeo`, `nftIdLeo`, `forgeTokenCancer`, `nftIdCancer`, `forgeTokenNeptune` and `nftIdNeptune` 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	Consider the above variables explicitly `immutable`.
Resolution	✓ RESOLVED

Issue #14	Outdated comment above `BONUS_MULTIPLIER`
Severity	Informational

Location	`Lines 39-40`
	`// Bonus muliplier for early virgo makers.`
	`uint256 public constant BONUS_MULTIPLIER = 100;`

Description	There is a comment above `BONUS_MULTIPLIER` that indicates that it is used for early Virgo makers; however, it is used for calculating the NFT-based bonus rewards. We thus believe this comment might be outdated.
Recommendation	Consider replacing this comment with a more accurate one to ease the process for third-party reviewers.
Resolution	✓ RESOLVED

PALADIN

BLOCKCHAIN SECURITY