PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For SpookyCrow Finance

24 November 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 SpookyCrow Finance on the Fantom Opera 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	SpookyCrow
URL	https://spookycrow.finance/
Platform	Fantom Opera
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
`SpookyCrowToken`	`0x026127430a616140e9411a8f0d83e6df8d31cd79`	MATCH
`MasterChef`	`0x9B35f3466C2Ec24d32c325Af9383Ff32303296F7`	MATCH

1.3 Findings Summary

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

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 SpookyCrowToken

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

1.3.2 MasterChef

ID	Severity	Summary	Status
02	LOW	`pendingCrow` will show inaccurate pending harvests on the dapp frontend if the pending rewards causes `totalSupply` to be exceed `MAXSUPPLYCAP`	ACKNOWLEDGED
03	INFO	Minting can be optimized to only mint if amount is non-zero to save gas consumption	ACKNOWLEDGED
04	INFO	Duplicate calculation of `devReward` in `updatePool`	ACKNOWLEDGED
05	INFO	State variables initialized in the constructor and never modified can be set to `immutable`	ACKNOWLEDGED

2 Findings

2.1 Token

The SpookyCrow token is a simple ERC-20 token which will be used as the main reward token for the Masterchef. It allows for SpookyCrow tokens to be minted when the mint function is called by the owner of the contract, which at the time of deployment would be the SpookyCrow team. Users should therefore carefully inspect that the ownership of the contract has been transferred to the Masterchef. The token has a maximum supply of 40,000.

2.1.1 Token Overview

Address	0x026127430a616140e9411A8f0d83E6dF8d31Cd79
Token Supply	25,000
Decimal Places	18
Transfer Max Size	No maximum
Transfer Min Size	No minimum
Transfer Fees	None
Pre-mints	200

2.1.2 Privileged Roles

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

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

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 200 tokens were pre-minted and ownership of the token has been transferred to the Masterchef.

2.2 MasterChef

The SpookyCrow Masterchef is a fork of Goose Finance's Masterchef. A notable feature of the Goose Masterchef is that Goose removed the migrator function, which can be used maliciously to steal user's tokens, when they forked Pancakeswap. SpookyCrow has limited the deposit fee to at most 4%. Emissions have been modified to use timestamps instead of block numbers.

2.2.1 Privileged Roles

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

- `add`
- `set`
- `updateEmissionRate`
- `updateStartTime`
- `transferOwnership`
- `renounceOwnership`

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

- `setDevAddress`

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

- `setFevAddress`

2.2.2 Issues & Recommendations

Issue #02	**`pendingCrow` will show inaccurate pending harvests on the dapp frontend if the pending rewards causes `totalSupply` to be exceed `MAXSUPPLYCAP`**
Severity	LOW SEVERITY
Description	Similarly to `updatePool`, `pendingCrow` does not check if the pending rewards will cause the total supply to exceed the `MAXSUPPLYCAP`. This can cause inaccurate pending harvests to be shown towards the end of token emissions.

Recommendation	Consider factoring in the `MAXSUPPLYCAP`, and set the pending reward to be the difference between `MAXSUPPLYCAP` and `totalSupply` if the pending reward causes `totalSupply` to exceed `MAXSUPPLYCAP`.
	
	`uint256 CrowReward = multiplier`
	`.mul(CrowPerSecond)`
	`.mul(pool.allocPoint)`
	`.div(totalAllocPoint);`
	
	`if (Crow.totalSupply().add(CrowReward) > Crow.maxSupply()) {`
	`CrowReward =`
	`Crow.maxSupply().sub(Crow.totalSupply());`
	`}`
	`accCrowPerShare =`
	`accCrowPerShare.add(CrowReward.mul(1e18).div(pool.lpSupply));`

Resolution	ACKNOWLEDGED

Issue #03	Minting can be optimized to only mint if amount is non-zero to save gas consumption
Severity	INFORMATIONAL

Location	Line 1454~:
	`if (totalRewards <= Crow.maxSupply()) {`
	`    // mint as normal as not at maxSupply`
	`    Crow.mint(devaddr, CrowReward.div(10));`
	`    Crow.mint(address(this), CrowReward);`
	`} else {`
	`    // mint the difference only to MC, update CrowReward`
	`    CrowReward = Crow.maxSupply().sub(Crow.totalSupply());`
	`    Crow.mint(address(this), CrowReward);`
	`}`
	
	`if (CrowReward != 0) {`
	`    // only calculate and update if CrowReward is non 0`
	`    pool.accCrowPerShare = pool.accCrowPerShare.add(`
	`        CrowReward.mul(1e18).div(pool.lpSupply)`
	`    );`
	`}`

Description	If the total supply of tokens has reached the `maxSupply`, 0 tokens will continue to be minted each time `updatePool` is called.

Recommendation	The following optimization can be done to only mint if the amount is non-zero:
	
	`if (totalRewards <= Crow.maxSupply()) {`
	`    // mint devReward as normal as not at maxSupply`
	`    Crow.mint(devaddr, CrowReward.div(10));`
	
	`} else {`
	`    // update CrowReward`
	`    CrowReward = Crow.maxSupply().sub(Crow.totalSupply());`
	`}`
	
	`if (CrowReward != 0) {`
	`    // only mint, calculate and update if CrowReward is non 0`
	`    Crow.mint(address(this), CrowReward);`
	`    pool.accCrowPerShare = pool.accCrowPerShare.add(`
	`        CrowReward.mul(1e18).div(pool.lpSupply)`
	`    );`
	`}`

Resolution	ACKNOWLEDGED

Issue #04	**Duplicate calculation of `devReward` in `updatePool`**
Severity	INFORMATIONAL

Location	Line 1449:
	`uint256 devReward = CrowReward.div(10);`
	`uint256 totalRewards = Crow.totalSupply().add(devReward).add(CrowReward);`
	`if (totalRewards <= Crow.maxSupply()) {`
	`  Crow.mint(devaddr, CrowReward.div(10));`

Description	`devReward` is calculated once when being used for calculating the `totalRewards`, but is calculated again when minting.
Recommendation	Use the devRewards variable in the `mint` function to avoid duplicate calculation.
Resolution	ACKNOWLEDGED

Issue #05	**State variables initialized in the constructor and never modified can be set to immutable**
Severity	INFORMATIONAL
Description	The following state variables can be set to immutable as they are never changed after initialization: • `Crow`
Recommendation	Add the immutable keyword to the above state variable(s).
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY