PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Space X (Round 2)

29 September 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 Space X on the Binance Smart Chain. 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	Space X
URL	https://farm.space/
Platform	Binance Smart Chain
Language	Solidity

1.2 Contracts Assessed

Name	Contract	Live Code Match
SpaceXChef	0xff3A7138055b285a195c9fcd92BB7B531266A472	MATCH

1.3 Findings Summary

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

ID	Severity	Summary	Status
01	HIGH	Uncapped withdrawal fees may steal all users' funds	RESOLVED
02	MEDIUM	distributeWeth may be called by any user to accidentally transfer their WETH	ACKNOWLEDGED
03	LOW	`accSpaceXPerShare` may be miscalculated when token maximum supply is almost reached	ACKNOWLEDGED
04	INFO	Contract should use the SafeERC20 library for approvals	ACKNOWLEDGED

2 Findings

2.1 SpaceXChef

This second report was commissioned by the client after having made several alterations to the previously-audited Masterchef contract, the more notable change being the addition of withdrawal fees. As a result, users wishing to stake their funds in the Masterchef may be met with both deposit and withdrawal fees, and the latter is payable in the `emergencyWithdraw` function as well. This second audit report was solely performed on the deployed Masterchef contract.

Deposit fees are no longer split and paid to the fee and vault addresses, but rather have been consolidated to only the `feeAddress`. The token minting for stakers has also been altered to impose a maximum token supply of 10 million tokens.

2.1.1 Privileged Roles

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

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

2.1.2 Issues & Recommendations

Issue #01	Uncapped withdrawal fees may steal all users' funds
Severity	HIGH SEVERITY
Description	Although there is an upper limit to `_withdrawalFeeBP` in the `add` function, there are no such safeguards in the `set` function. As a result, a malicious owner may steal users' funds via the following set of methods: 1. Add a new pool to the Masterchef with up to 10% withdrawal fees. 2. Set `_withdrawalFeeBP` to 100%. 3. All users who call either `withdraw` or `emergencyWithdraw` would then lose all funds.
Recommendation	Consider adding in the following safeguard to limit withdrawal fees to 10% in the set function: `require(_withdrawalFeeBP <= 1000, "add: invalid deposit fee basis points");` Alternatively, as the contract has already been deployed, the use of a `safeOwner` wrapper contract to limit withdrawal fees using the same safeguard above would work just as well.
Resolution	RESOLVED The client has transferred ownership of the Masterchef contract to a SafeSpaceXTimelock contract, and then ownership of contract is a SafeTimelockAdmin contract. Although somewhat confusing and lengthy, these contracts now limit setting withdrawal fees to at most 1%.

Issue #02	**`distributeWeth` may be called by any user to accidentally transfer their WETH**
Severity	MEDIUM SEVERITY
Description	The `distributeWeth` function should be restricted to only being called by privileged addresses, such as the owner or whitelisted addresses. Currently, if any user has given WETH allowance to the Masterchef (by default, infinite approval is given when users' approve a WETH pool), then they may accidentally call `distributeWeth` to inadvertently transfer their WETH tokens to the Masterchef for distribution rather than depositing into a pool.
Recommendation	Consider only allowing certain privileged addresses such as the owner or other whitelisted addresses to call this function, to prevent users from accidentally transferring their WETH to the Masterchef for distribution.
Resolution	ACKNOWLEDGED

Issue #03	**accSpaceXPerShare may be miscalculated when token maximum supply is almost reached**
Severity	LOW SEVERITY

Location	Lines 230-233
	`uint256 spaceXReward = multiplier.mul(spaceXPerBlock).mul(pool.allocPoint).div(totalAllocPoint);`
	`mintCapped(devAddress, spaceXReward.div(10));`
	`mintCapped(address(this), spaceXReward);`
	`pool.accSpaceXPerShare = pool.accSpaceXPerShare.add(spaceXReward.mul(1e18).div(lpSupply));`

Description	In the `updatePool` function, `pool.accSpaceXPerShare` is increased by `spaceXReward`. In the `mintCapped` function, however, if total token supply has almost been reached, then it will mint less than `spaceXReward`. The following illustrates this point: 1. Assume that: a. totalSupply = 9,999,999 and MAX_SPACEX_SUPPLY = 10,000,000. b. `spaceXReward` = 5 2. `mintCapped` is called, and since `if(totalSupply.add(amount) > MAX_SPACEX_SUPPLY)` is `false`, 0.5 tokens are minted to `devAddress`. a. `totalSupply` = 9,999,999.5 3. `mintCapped` is called again to mint tokens to the Masterchef contract, and now `if(totalSupply.add(amount) > MAX_SPACEX_SUPPLY)` returns `true`, so only 0.5 tokens are minted to the Masterchef. 4. `pool.accSpaceXPerShare` is incremented by `spaceXReward` = 5 although only 1 was actually minted, resulting in `pool.accSpaceXPerShare` increasing more than it should.

Recommendation

To fix this, `mintCapped` has to return the amount it actually minted, and `updatePool` is to use that to calculate the new `accSpaceXPerShare`.

```
function mintCapped(address to, uint256 amount) private
returns (uint256) {
    uint256 totalSupply = spaceXToken.totalSupply();
    if (totalSupply ==  MAX_SPACEX_SUPPLY) {
        return 0;
    }

    if(totalSupply.add(amount) > MAX_SPACEX_SUPPLY) {
        amount = MAX_SPACEX_SUPPLY.sub(totalSupply);
    }
    spaceXToken.mint(to, amount);
    return amount;
}
```

And then in `updatePool`:

```
mintCapped(devAddress, spaceXReward.div(10));
uint256 mintedToMC = mintCapped(address(this),
spaceXReward);
if (mintedToMC != 0) {
        pool.mintCapped =
pool.accSpaceXPerShare.add(mintedToMC.mul(1e18).div(lpSupply
));
}
```

Resolution

ACKNOWLEDGED

Issue #04	Contract should use the SafeERC20 library for approvals
Severity	INFORMATIONAL
Description	Currently the contract sets approvals via `lpToken.approve`. It is generally considered best practice to use OpenZeppelin's SafeERC20 library to set approvals instead.
Recommendation	Consider using SafeERC20's `safeIncreaseAllowance` and `safeDecreaseAllowance` instead.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY