PALADIN
BLOCKCHAIN SECURITY

# Smart Contract Security Assessment

Final Report

## For Sandman Farm

18 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 Sandman Farm. 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.

The Sandman contracts were forked from one of our audited projects, PolyWantsACracker, with minimal changes. Therefore the contracts are a tested, secure set of contracts.

## 1.1 Summary

| | |
|---|---|
| **Project Name** | Sandman Farm |
| **URL** | https://sandman.farm/ |
| **Platform** | Polygon |
| **Language** | Solidity |

## 1.2 Contracts Assessed

As the client is planning a stealth launch, we have redacted the full contract address at his request. We will update this report once the launch has been completed.

| Name | Contract | Live Code Match |
|---|---|---|
| MorpheusToken | ...Ffe7CB | MATCH |
| SandManToken | ...fF6A75 | MATCH |
| MasterChef | ...C6E6A1 | MATCH |
| MorpheusDream | ...cb3679 | MATCH |
| Timelock | ...B43d41 | MATCH |

## 1.3 Findings Summary

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

No issues found.

### 1.3.2 SandmanToken

No issues found.

### 1.3.3 MasterChef

| ID | Severity | Summary | Status |
|---|---|---|---|
| 01 | LOW | The `pendingSandman` function will revert if `totalAllocPoint` is zero | RESOLVED |
| 02 | INFORMATIONAL | `sandman` can be renamed and made `immutable` | RESOLVED |

### 1.3.4 MorpheusDream

| ID | Severity | Summary | Status |
|---|---|---|---|
| 03 | INFORMATIONAL | `morpheusToken` and `sandManToken` can be made immutable | RESOLVED |

### 1.3.5 Timelock

No issues found.

# 2 Findings

## 2.1 MorpheusToken

The MorpheusToken contract is forked from PolyWantsACracker's presale contract. The Morpheus token is the presale token that can be swapped to Sandman tokens via the MorpheusDream contract at a 1:1 ratio. 30,000 presale tokens are pre-minted and priced at $5 each, payable via USDC. The presale duration lasts for 179,800 blocks, which is just over 5 days (as block times are non-constant on Polygon, this duration may fluctuate).

### 2.1.1 Privileged Roles

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

- `setStartBlock`

### 2.1.2 Issues & Recommendations

No issues found.

## 2.2 SandManToken

The contract allows for Sandman tokens to be minted when the mint function is called by Owner, who at the time of deployment would be the deployer, though ownership is generally transferred to the Masterchef via the `transferOwnership` function for emission rewards to be minted and distributed to users staking in the Masterchef. The `mint` function can be used to pre-mint tokens for various uses including injection of initial liquidity, token presale, airdrops, and others.

On deployment, 42,000 tokens were minted to the deployer address. We also confirm that the token owner has been transferred to the Masterchef, and that the project team is no longer able to manually call the `mint` function.

### 2.2.1 Token Overview

| | |
|---|---|
| **Address** | `...fF6A75` |
| **Token Supply** | 100,000 (one hundred thousand) |
| **Decimal Places** | 18 |
| **Transfer Max Size** | No maximum |
| **Transfer Min Size** | No minimum |
| **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

No issues found.

## 2.3 MasterChef

The Sandman Masterchef contract was forked from PolyWantsACracker, which was previously audited by Paladin. As such, it is a secure Masterchef contract and we commend Sandman on forking an audited, proven Masterchef. Deposit fees have an upper limit of 4.01%, transfer tax tokens are properly accounted for, and the notorious `migrator` function has also been removed.

A notable feature of this Masterchef is that the maximum token supply of 100,000 tokens is enforced in the `updatePool` function, which halts minting should the maximum supply be reached. Additionally, the use of Solidity version 0.8.3 means that overflow checks are built-in.

### 2.3.1 Privileged Roles

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

- `add`
- `set`
- `setFeeAddress`
- `setStartBlock`

### 2.3.2 Issues & Recommendations

| Issue #01 | **The `pendingSandman` function will revert if `totalAllocPoint` is zero** |
|---|---|
| **Severity** | LOW SEVERITY |
| **Description** | In the `pendingSandman` function, at some point a division is made by the `totalAllocPoint` variable. If all pools have their rewards set to zero, this variable will be zero as well. The requests will then revert with a division by zero error. |
| **Recommendation(s)** | Consider only calculating the accumulated rewards since the `lastRewardBlock` if the `totalAllocPoint` variable is greater than zero.<br><br>This check can simply be added to the existing check that verifies the `block.number` and `lpSupply`, like so:<br><br>`if (block.number > pool.lastRewardBlock && lpSupply != 0 && totalAllocPoint > 0) { … }` |
| **Resolution** | RESOLVED |

| Issue #02 | **`sandman` 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. |
| **Recommendation(s)** | Consider making sandman explicitly immutable. |
| **Resolution** | RESOLVED |

## 2.4 MorpheusDream

This contract allows Morpheus token holders to swap their presale tokens to Sandman tokens at a 1:1 ratio. The Morpheus tokens are burned, whilst any excess (unredeemed) Sandman tokens may also be burned by the Owner. In order for the swaps to take place, Sandman tokens have to be transferred to this swap contract.

### 2.4.1 Privileged Roles

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

- `sendUnclaimedSandManToDeadAddress`
- `setStartBlock`

### 2.4.2 Issues & Recommendations

| Issue #03 | `morpheusToken` and `sandManToken` 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. |
| **Recommendation(s)** | Consider making `morpheus`, `sandmanAddress` explicitly `immutable`. |
| **Resolution** | RESOLVED<br>The presale contract has already been deployed and will no longer be in further use. |

## 2.5 Timelock

The Timelock contract is a clean fork of Compounder Finance's timelock. This is the most common contract used in DeFi to time lock governance access and is thus compatible with most third-party tools.

| Parameter | Value | Description |
|---|---|---|
| **Delay** | 6 hours | The `delay` indicates the time the administrator has to wait after queuing a transaction to execute it. |
| **Minimum Delay** | 6 hours | The `minDelay` indicates the lowest value that the `delay` can minimally be set.<br><br>Sometimes, projects will queue a transaction that sets the `delay` to zero with the hope that nobody notices it. However, because of the minimum delay parameter, the value of `delay` can never be lower than that of the `minDelay` value. Note that the administrator could still queue a transaction to simply transfer the ownership back to their own account so it is still important to inspect every transaction carefully. |
| **Grace Period** | 14 days | After the delay has expired after queueing a transaction, the administrator can only execute it within the grace period. This is to prevent them from hiding a malicious transaction among much earlier transactions, hoping that it goes unnoticed or buried, which can be executed in the future. |

### 2.5.1 Issues & Recommendations

No issues found.


PALADIN
BLOCKCHAIN SECURITY