CERTIK

# Security Assessment

## MYSTERYBIRD

May 23rd, 2021

# Table of Contents


## Summary

## Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

## Findings

- BCK-01 : Constant value could be declared as a variable
- BCK-02 : Comparison with literal `true`
- BCK-03 : Missing check for parameter and emit event
- BCK-04 : Old compiler version declaration and version not locked
- CCK-01 : Missing emit event
- FLC-01 : Missing zero address validation
- FLC-02 : Missing emit events
- FLC-03 : Potentially excessive permissions
- FLC-04 : Missing parameters check
- FLC-05 : Constant value could be declared as a variable
- FLC-06 : Check effect interaction pattern violated
- MDX-01 : Comparison with literal `true`
- SAT-01 : Function naming doesn't match the operating environment
- SAT-02 : Possible to gain ownership after renouncing the contract ownership
- SAT-03 : State variables that could be declared constant
- SAT-04 : State variables that could be declared immutable
- SAT-05 : Divide before multiple

## Formal Verification Requests

## Appendix

## Disclaimer

## About

# Summary

This report was written for the MYSTERYBIRD smart contract to discover issues and vulnerabilities in its smart contract source code, as well as any contract dependencies that are not part of an officially recognized library. A comprehensive inspection was carried out using static analysis and manual review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

# Overview

## Project Summary

| | |
|---|---|
| Project Name | MYSTERYBIRD |
| Description | Mystic Bird Finance is a leveraged income farm protocol based on Binance Smart Chain (BSC) released by Mystic Bird Financial Labs. |
| Platform | BSC |
| Language | Solidity |
| Codebase | https://www.mysterybird.net/?shareUser |
| Commits | 3ad563ad05bc7156d9fbf3f7731b244be04528ea |

## Audit Summary

| | |
|---|---|
| Delivery Date | May 23, 2021 |
| Audit Methodology | Static Analysis, Manual Review |
| Key Components | |

## Vulnerability Summary

| | |
|---|---|
| Total Issues | 17 |
| Critical | 0 |
| Major | 0 |
| Medium | 0 |
| Minor | 1 |
| Informational | 16 |
| Discussion | 0 |

# Audit Scope

| ID | file | SHA256 Checksum |
| --- | --- | --- |
| BCK | Bank.sol | 374347a43eb3a5ae27e53e0e5cddfb5632fc7feb9d15ff95696b2fb2a7bba378 |
| CCK | Conf.sol | 6e173eeffbac7f36807b0b91269179200834e10366d8330cac9f65edb702c972 |
| DTC | DebtToken.sol | 1f56ff6de8fc3e1306f37a0f198942b26b0d5b411b72d14d962405ce17bb1301 |
| ASE | ESPStrategy/espAddStrategy.sol | c5538166423c7c5df5c31934c77d12436bb6492aaa367348004799c4d9666029 |
| LSE | ESPStrategy/espLiquidateStrategy.sol | 445301e459e3efc01eb6636e8d910c94cc1ff2ddc1d01ed1bddb3be15506525f |
| WSE | ESPStrategy/espWithdrawStrategy.sol | 5054929c5b26202fd62611310fd4d861f46e9372312595854368a60532b0c3f8 |
| FLC | FairLaunch.sol | dd4bec584383297bb5c2d46ac90eac8704cd3eb2b509e8190f8b03102fa5ca15 |
| MDX | MDXGoblin.sol | ef8ee44b4423d99cb010a5851799184c027017eef41667fdc1005069eb8cba58 |
| PGC | PancakeGoblin.sol | 75fc3012aabfdce1f7fa7a5235f535d9b285f9f1ca3b977122c422aaba440b83 |
| RTC | RabbitToken.sol | f2519130dffa5d73c957c7feaacacabd909135189d0c687220d1711a87fdae46 |
| SAT | StrategyAddTwoSidesOptimal.sol | 4cfa2ccf91bcfa19dc3f8394862116491f7e52d2ca460fb02f031fa471a7bf1d |
| SAO | StrategyAllTokenOnly.sol | 5f02eed9f968f0f5e211196f58501ef8e1d6d32d081722dfe99412efc84f687f |
| SLC | StrategyLiquidate.sol | af8b5364f250302f94fec9828985a916e395d4aa1517e22e29c98d8212f96dc2 |
| SWM | StrategyWithdrawMinimizeTrading.sol | 3338c22301192918333d5ffa4d05c09d6fe5f67f9cb62b2fc30d3d3a4000ae03 |

# Centralization Roles

The smart contract introduces several authorizations.

## Governance:

[Bank.sol]

- updateConfig()
- createProduction()
- withdrawReserve()
- createkillWhitelist()

[Conf.sol]

- setParams()

[FairLaunch.sol]

- renounceOwnership()
- transferOwnership()
- setRabbitPerBlock()
- setBonus()
- manualMint()

## Strategist:

[StrategyAddTwoSidesOptimal.sol]

- execute()
- swapMiningReward()
- recover()

[StrategyAllTokenOnly.sol]

- execute()
- swapMiningReward()
- recover()

[StrategyLiquidate.sol]

- execute()

# Centralization Roles

The smart contract introduces several authorizations.

## Governance:

[Bank.sol]

- updateConfig()
- createProduction()
- withdrawReserve()
- createkillWhitelist()

[Conf.sol]

- setParams()

[FairLaunch.sol]

- renounceOwnership()
- transferOwnership()
- setRabbitPerBlock()
- setBonus()
- manualMint()

## Strategist:

[StrategyAddTwoSidesOptimal.sol]

- execute()
- swapMiningReward()
- recover()

[StrategyAllTokenOnly.sol]

- execute()
- swapMiningReward()
- recover()

[StrategyLiquidate.sol]

- execute()

- swapMiningReward()
- recover()

## Controller:

[Bank.sol]

- addBank()

[FairLaunch.sol]

- addPool()
- setPool()

# Findings


| | |
|---|---|
| Critical | 0 (0.00%) |
| Major | 0 (0.00%) |
| Medium | 0 (0.00%) |
| Minor | 1 (5.88%) |
| Informational | 16 (94.12%) |
| Discussion | 0 (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| BCK-01 | Constant value could be declared as a variable | Gas Optimization | Informational | Resolved |
| BCK-02 | Comparison with literal `true` | Gas Optimization | Informational | Resolved |
| **BCK-03** | Missing check for parameter and emit event | **Centralization / Privilege** | **Informational** | **Resolved** |
| BCK-04 | Old compiler version declaration and version not locked | Language Specific | Informational | Resolved |
| **CCK-01** | Missing emit event | **Centralization / Privilege** | **Informational** | **Resolved** |
| FLC-01 | Missing zero address validation | Control Flow | Informational | Resolved |
| **FLC-02** | Missing emit events | **Centralization / Privilege** | **Informational** | **Resolved** |
| **FLC-03** | Potentially excessive permissions | **Centralization / Privilege** | **Minor** | **Resolved** |
| FLC-04 | Missing parameters check | Logical Issue | Informational | Resolved |
| FLC-05 | Constant value could be declared as a variable | Gas Optimization | Informational | Resolved |
| FLC-06 | Check effect interaction pattern violated | Logical Issue | Informational | Resolved |
| MDX-01 | Comparison with literal `true` | Gas Optimization | Informational | Resolved |

| ID | Title | Category | Severity | Status |
| --- | --- | --- | --- | --- |
| SAT-01 | Function naming doesn't match the operating environment | Coding Style | Informational | Acknowledged |
| **SAT-02** | Possible to gain ownership after renouncing the contract ownership | **Centralization / Privilege** | **Informational** | **Resolved** |
| SAT-03 | State variables that could be declared constant | Gas Optimization | Informational | Resolved |
| SAT-04 | State variables that could be declared immutable | Coding Style | Informational | Resolved |
| SAT-05 | Divide before multiple | Mathematical Operations | Informational | Acknowledged |

# BCK-01 | Constant value could be declared as a variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | Bank.sol: 1000, 1017, 1026, 1141 | Resolved |

## Description

We see the `10000` value is used many times in some contracts such as `Bank`.

```
require(health.mul(production.liquidateFactor) < debt.mul(10000), "can't liquidate");
```

## Recommendation

We recommend declaring those values as constant variables in all of the contracts.

## Alleviation

The Mystery Bird team heeded our advice and declared those values as constant variables.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## BCK-02 | Comparison with literal `true`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | Bank.sol: 1008 | Resolved |

### Description

The `killWhitelist[msg.sender] == true` code performs comparison with a boolean literal `true` which can be replaced with the negation of the expression to increase the legibility of the codebase.

### Recommendation

We recommend using the expression inside the require statement instead of comparison with boolean.

### Alleviation

The Mystery Bird team heeded our advice and use the expression inside the `require` statement.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# BCK-03 | Missing check for parameter and emit event

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Informational | Bank.sol: 1176 | Resolved |

## Description

Function `createkillWhitelist` is used to change the whitelist of kills, the parameter of `addr` should be checked for zero address. And it is better to add an event for tracking the changing whitelist.

```
function createkillWhitelist(address addr,bool status) external onlyGov {
  killWhitelist[addr] = status;
}
```

## Recommendation

We recommend adding `require(addr != address(0)` in the function. Add event for this function, and emit it.

## Alleviation

The Mystery Bird team took our suggestion and revised the code.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# BCK-04 | Old compiler version declaration and version not locked

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Language Specific | Informational | Bank.sol: 1 | Resolved |

## Description

`solc` frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

The contract uses some different versions, such as `pragma solidity ^0.5.16;`, `pragma solidity ^0.6.6;` and `pragma solidity ^0.5.0;`, and all of these are not locked. This is not recommended. Pragmas should be locked to specific compiler versions and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

## Recommendation

Avoid a floating pragma version (i.e. `pragma solidity ^0.5.0;`) instead specify pragma version without using the caret symbol, i.e., `pragma solidity 0.6.11;`.

Deploy with any of the following Solidity versions:

- 0.5.11 - 0.5.13,
- 0.5.15 - 0.5.17,
- 0.6.8,
- 0.6.10 - 0.6.11. Use a simple pragma version that allows any of these versions.

We recommend using the latest version of Solidity for testing.

## Alleviation

The Mystery Birds team took our advice and revised the latest version.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# CCK-01 | Missing emit event

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Informational | Conf.sol: 101~105 | Resolved |

## Description

Function `setParams` is used to change the configuration of Mystery, it is bestter to add event and emit it in this function.

## Recommendation

We recommend adding event and emit it in the function `setParams`.

## Alleviation

The Mystery Birds team took our advice and added an event to the feature.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# FLC-01 | Missing zero address validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Control Flow | Informational | FairLaunch.sol: 715 | Resolved |

## Description

Detect missing zero address validation.

## Recommendation

We recommend adding validation for address.

```
function setDev(address _devaddr) public {
  require(msg.sender == devaddr, "dev: wut?");
  require(_devaddr != address(0));
  devaddr = _devaddr;
}
```

## Alleviation

The Mystery Bird team heeded our advice and added validation for address.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# FLC-02 | Missing emit events

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| **Centralization / Privilege** | **Informational** | FairLaunch.sol: 715, 720, 726 | **Resolved** |

## Description

Several sensitive actions are defined without event declarations.

## Recommendation

We recommend adding events for sensitive action, and emit them in the functions.

## Alleviation

The Mystery Bird team heeded our advice and added events to the sensitive functions.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## FLC-03 | Potentially excessive permissions

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Minor | FairLaunch.sol: 785 | Resolved |

### Description

Function `manualMint` is only called by the owner, and it allows the caller to mint Rabbit to any recipient. To improve the trustworthiness of the project, any plan to mint token is better to move to the execution queue of the Timelock and also add an `emit event`, or make the owner Multi-sig.

```
function manualMint(address _to, uint256 _amount) public onlyOwner {
  IRabbit(address(rabbit)).mint(_to, _amount);
}
```

### Recommendation

We recommend adding an `emit event` at function `manualMint`. And it should transfer the owner of this contract to `Timelock`, or make the owner Multi-sig.

### Alleviation

The Mystery Birds team heeded our advice and added an event to the function.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## FLC-04 | Missing parameters check

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | FairLaunch.sol: 790 | Resolved |

### Description

The parameters of function `getMultiplier` are missing check.

### Recommendation

We recommend adding `require` check for the input parameters.

```
require(_lastRewardBlock <= _currentBlock, "error message");
```

### Alleviation

The Mystery Bird team heeded our advice and added a check fou the parameters.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# FLC-05 | Constant value could be declared as a variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | FairLaunch.sol: 812 | Resolved |

## Description

We see the `1e12` value is used many times in some contracts such as `FairLaunch`.

```
user.rewardDebt = user.amount.mul(pool.accRabbitPerShare).div(1e12);
user.bonusDebt = user.amount.mul(pool.accRabbitPerShareTilBonusEnd).div(1e12);
```

## Recommendation

We recommend declaring those values as constant variables in all of the contracts.

## Alleviation

The Mystery Bird team heeded our advice and declared those values as constant variables.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# FLC-06 | Check effect interaction pattern violated

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | FairLaunch.sol: 896~898 | Resolved |

## Description

The order of external call/transfer and storage manipulation must follow check effect interaction pattern.

## Recommendation

We recommend inheriting `ReentrancyGuard` contract and adding `nonReentrancy` modifier.

## Alleviation

The Mystery Bird team heeded our advice snd inherited `ReentrancyGuard` and added `nonReentrancy` modifier.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## MDX-01 | Comparison with literal `true`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | MDXGoblin.sol: 1013 | Resolved |

### Description

The `killWhitelist[msg.sender] == true` code performs a comparison with a boolean literal `true` that can be replaced with the negation of the expression to increase the legibility of the codebase.

### Recommendation

We recommend using the expression inside the require statement instead of comparison with boolean.

### Alleviation

The Mystery Bird team heeded our advice and used the expression inside the require statement.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## SAT-01 | Function naming doesn't match the operating environment

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | StrategyAddTwoSidesOptimal.sol: 291 | Acknowledged |

### Description

The Mystery Bird contract uses Pancakeswap for swapping and adds liquidity to the Pancakeswap pool but naming it Uniswap. Function `swapTokensForEth(uint tokenAmount)` swaps Mystery Bird token for `BNB` instead of `ETH`.

### Recommendation

Change `Uniswap` and `ETH` to `Pancakeswap` and `BNB` in the contract respectively to match the operating environment and avoid confusion.

### Alleviation

The Mystery Bird team has acknowledged this finding.

## SAT-02 | Possible to gain ownership after renouncing the contract ownership

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Informational | StrategyAddTwoSidesOptimal.sol: 56~59 | Resolved |

## Description

An owner is possible to gain ownership of the contract even if he calls function `renounceOwnership` to renounce the ownership. This can be achieved by performing the following operations:

1. Call `lock` to lock the contract. The variable `_previousOwner` is set to the current owner.
2. Call `unlock` to unlock the contract.
3. Call `renounceOwnership` to leave the contract without an owner.
4. Call `unlock` to regain ownership.

## Recommendation

We advise updating/removing `lock` and `unlock` functions in the contract; or removing the `renounceOwnership` if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound finance as reference. Reference: https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol

## SAT-03 | State variables that could be declared constant

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | StrategyAddTwoSidesOptimal.sol: 635 | Resolved |

### Description

Constant state variables could be declared constant to save gas. And constant variable should be named UPPER_CASE_WITH_UNDERSCORES. And there are similar state variables in the contract `StrategyAllTokenOnly` line 648, `StrategyLiquidate` line 623, and `StrategyWithdrawMinimizeTrading` line 621.

```
address public wbnb = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c;
```

### Recommendation

We recommend declaring the state variables as constant variables. And constant variables should be named UPPER_CASE_WITH_UNDERSCORES.

### Alleviation

The Mysterty Bird team heeded our advice and declared the state variables as constant variables.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

# SAT-04 | State variables that could be declared immutable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | StrategyAddTwoSidesOptimal.sol: 636 | Resolved |

## Description

The immutable state variables could be declared `immutable`.

## Recommendation

We recommend declaring the state variable `immutable`.

## Alleviation

The Mystery Bird team heeded our advice and declared the state variables `immutable`.

The recommendations were applied in commit 0567922d6faf84dbb7d05c86562993a280c40e80.

## SAT-05 | Divide before multiple

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Mathematical Operations | Informational | StrategyAddTwoSidesOptimal.sol: 694 | Acknowledged |

### Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision. And there are similar codes in the contracts `espAddStrategy` line 700 and `StrategyAllTokenOnly` line 695.

```
uint256 c = _c.mul(1000).div(amtB.add(resB)).mul(resA);
```

### Recommendation

We recommend ordering multiplication before division.

### Alleviation

The Mystery Bird team has acknowledged this.

# Appendix

## Finding Categories

### Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

### Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

### Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

## Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

# About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.