CERTIK

Security Assessment

Bird Finance

Apr 3rd, 2021

Summary

This report has been prepared for Bird Finance smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, 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 10 findings that ranged from major 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	Bird Finance
Description	
Platform	Heco
Language	Solidity
Codebase	https://github.com/BirdFinance/bird-core/commits/audit
Commits	1. 1387009f7e267ac5540fa3e10f66dc54b951311a 2. 5f88fb01732b0c1be3448544f6b57696a082b6ce

Audit Summary

Delivery Date	Apr 03, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

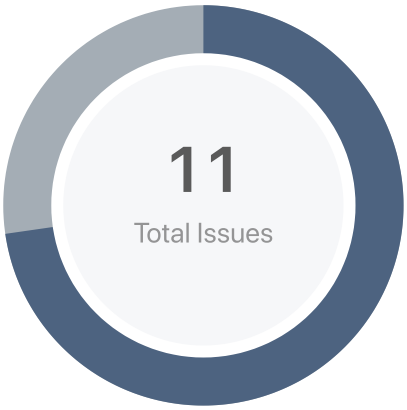
Vulnerability Summary

Total Issues	11
Critical	0
Major	0
Minor	3
Informational	8
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
BTN	BirdToken.sol	5e6fbc60251dc5368e63d90126163dea51aa02b2cff8d10d32277ec6f90e974b
LPT	LPTokenWrapper.sol	d4238bf7a738f08f09c1a473f36452b01a04a977b99d11e771918764c473ae85
POO	Pool.sol	a142589981869d147daeb10f9ac849b047ae21d38f3e79d31df5ca36c4ca758f

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	3 (27.27%)
Informational	8 (72.73%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
BTN-1	Missing Emit Events	Optimization	Informational	Declined
BTN-2	Proper Usage of `public` And `external` Type	Optimization	Informational	Declined
BTN-3	Typo	Logical Issue	Informational	Resolved
BTN-4	Redundant Code	Logical Issue	Informational	Resolved
BTN-5	Incorrect Value Read	Logical Issue	Minor	Resolved
POO-1	Lack of Input Validation	Volatile Code	Informational	Resolved
POO-2	Implementation of `transferBack()`	Optimization	Minor	Resolved
POO-3	Missing Emit Events	Optimization	Informational	Declined
POO-4	Initialization of `Pool`	Optimization	Informational	Declined
POO-5	Check the Balance of Reward Token	Logical Issue	Minor	Declined
POO-6	Reward of Inviter	Logical Issue	Informational	Declined

BTN-1 | Missing Emit Events

Category	Severity	Location	Status
Optimization	Informational	BirdToken.sol: 173~205	Declined

Description

Several key actions are defined without event declarations.

Recommendation

Consider emitting events for key actions.

Alleviation

No alleviation.

BTN-2 | Proper Usage of `public` And `external` Type

Category	Severity	Location	Status
Optimization	Informational	BirdToken.sol: 76~90, 97~153, 161~168, 184~190, 207~210, 292~294	Declined

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than `public` functions.

Recommendation

Consider using the `external` attribute for functions never called from the contract.

Alleviation

No alleviation.

BTN-3 | Typo

Category	Severity	Location	Status
Logical Issue	Informational	BirdToken.sol: 171	Resolved

Description

The error message in `require(_isExcluded[account], "Account is already excluded")` of function `includeInReward()` does not describe the error correctly.

Recommendation

Consider changing the message `"Account is already excluded"` to `"Account is already included"`.

Alleviation

The development team heeded our advice and resolved this issue in commit 5f88fb01732b0c1be3448544f6b57696a082b6ce.

BTN-4 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	BirdToken.sol: 395~397	Resolved

Description

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in else.

Recommendation

The following code can be removed:

```
...
else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
   _transferStandard(sender, recipient, amount);
}
...
```

Alleviation

The development team heeded our advice and resolved this issue in commit 5f88fb01732b0c1be3448544f6b57696a082b6ce.

BTN-5 | Incorrect Value Read

Category	Severity	Location	Status
Logical Issue	Minor	BirdToken.sol: 347~351	Resolved

Description

In the function `swapAndLiquify()`, value `address(this).balance` i.e. balance of `ETH` is assigned to `initialBalance` , however, what swaps for in the function `swapTokensForEth()` is `WHT`.

The name of function `swapTokensForEth()` does not describle what it does correctly.

Recommendation

Consider querying the balance of `WHT` instead of `ETH`, in the function `swapAndLiquify()`.

Consider rename the function `swapTokensForEth()` to `swapTokensForHt()`.

Alleviation

The development team responsed that the smart contract is universally adapted to all Ethereum-family blockchains and maybe deployed to more blockchains further, although the native token/currency is HT on Heco. The term Eth in function name swapTokensForEth() is as above, not designated for native currency, which is Eth of Ethereum. For now, the smart contract is planed to be deployed on Heco.

POO-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	Pool.sol: 74~77	Resolved

Description

The assigned address of `bird`, `lpt` and `minerOwner` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in constructor of contract `Pool.sol`. Violation of this may cause errors.

Recommendation

Check that the address is not zero by adding following checks in the constructor of contract `Pool.sol`, like:

```
require(bird_ != address(0), "bird_ is zero address");
require(lptoken_ != address(0), "lptoken_ is zero address");
require(minerOwner_ != address(0), "minerOwner_ is zero address");
```

Alleviation

The development team heeded our advice and resolved this issue in commit 5f88fb01732b0c1be3448544f6b57696a082b6ce.

POO-2 | Implementation of `transferBack()`

Category	Severity	Location	Status
Optimization	Minor	Pool.sol: 158~160	Resolved

Description

Function `transferBack()` can transfer bird token to anyone by the owner.

Recommendation

Consider emitting event for the function, or changing it to a recovery function in order to recovery ERC20 tokens which are transferred to this pool by mistake, like:

```
event Recovered(address token, address back, uint256 amount);

  function recoverERC20(address _token, uint256 _amount)
  external
  onlyOwner()
  {
      require(_token != address(lpt), "Cannot withdraw staking tokens");

      IERC20(_token).safeTransfer(back, _amount);
      emit Recovered(_token, _amount);
  }
```

Alleviation

The development team heeded our advice and resolved this issue in commit 5f88fb01732b0c1be3448544f6b57696a082b6ce.

POO-3 | Missing Emit Events

Category	Severity	Location	Status
Optimization	Informational	Pool.sol: 165~191	Declined

Description

Several key actions are defined without event declarations.

Recommendation

Consider emitting events for key actions.

Alleviation

No alleviation.

POO-4 | Initialization of `Pool`

Category	Severity	Location	Status
Optimization	Informational	Pool.sol: 162~173	Declined

Description

Function `initSet()` can be called more than once, is that designed as expected?

Recommendation

Consider adding logical to prevent repeated call on the function, like applying modifier `initializer` of contract `Initializable` of `openzepplin`.

Alleviation

No alleviation.

POO-5 | Check the Balance of Reward Token

Category	Severity	Location	Status
Logical Issue	Minor	Pool.sol: 175~191	Declined

Description

Function `updateRewardRate()` updates the reward distributed to accounts but missing check whether the balance of reward token is enough or not.

Recommendation

Consider checking the balance of reward token is enough or not to distribute the reward.

Alleviation

No alleviation.

POO-6 | Reward of Inviter

Category	Severity	Location	Status
Logical Issue	Informational	Pool.sol:	Declined

Description

The feature of inviter and invitee are involved in the commit 5f88fb01732b0c1be3448544f6b57696a082b6ce.

- Reward of inviter is not a part of reward of invitee, but a percentage of reward of invitee.
- Default inviter will get reward for accounts who do not set their inviter.

Do the above be designed as expected?

Alleviation

The development team responsed as the below: Assume the invitee will have received 1000 tokens and at the preset ratio the inviter will have received 10 tokens, thus the total yield is 110 tokens. If the invitee assigned some address as inviter, that address should get 10 tokens. If not, the 10 tokens bonus will belongs to a default address.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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 exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that 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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

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 and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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.