CERTIK

Security Assessment

CherrySwap

Apr 8th, 2021

Summary

This report has been prepared for CherrySwap 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 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	CherrySwap
Description	CherrySwap is a DeFi project developed based on OKExchain.
Platform	OKExChain
Language	Solidity
Codebase	https://github.com/cherryswapnet/swap-farm/tree/audit https://github.com/cherryswapnet/swap-core/tree/audit https://github.com/cherryswapnet/swap-periphery/tree/audit
Commits	1. e3d6d8b4ac76f8775bab28151bd47738548b8ed5 2. 851c0e4163a4878f8389025364f7eaa2e9c30d1d 3. 6915487f05bfa692381062cfe139430f72a9b4d3

Audit Summary

Delivery Date	Apr 08, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

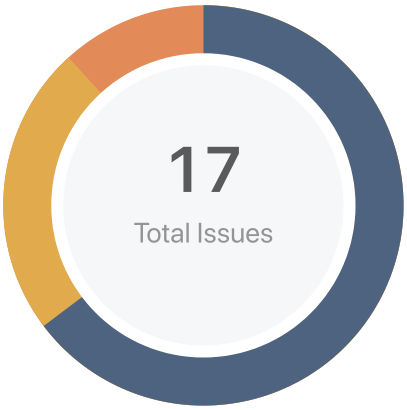
Vulnerability Summary

Total Issues	17
Critical	0
Major	2
Minor	4
Informational	11
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CER	swap-core-audit/contracts/CherryERC20.sol	c11492fd2094a88b35015c9ae19822494f67a8f61c955b52c78aec6e42411016
CFT	swap-core-audit/contracts/CherryFactory.sol	20df75ffdf2e81bc117c583a0c4664e0cdd6762fe4a834d541f9b127682e0b6c
CPT	swap-core-audit/contracts/CherryPair.sol	5511b803956298dc07ceb6605fd9973321d9485842b126ae23c06b56e1d6ad1c
IFO	swap-farm-audit/contract/IFO.sol	76c4ff23008a3cba9330c94e0b96304290511bb8fd6f07914b6a0732e92ec91b
IFB	swap-farm-audit/contract/IFOByProxy.sol	59b5aef70a7adbbd7e4681d4163687a04453cb52977bf5e0aaeaec857c57e1a2
CMT	swap-periphery-audit/contracts/CherryMigrator.sol	02ffdebe88d2f0d915af0d5e30b058d705488846a42f4bd571de5facfccd0912
CRT	swap-periphery-audit/contracts/CherryRouter.sol	5dd0247c06df11bc974f59ffe208b12ee983594d641003659521fb609a1e82b1
CRI	swap-periphery-audit/contracts/CherryRouter01.sol	5f1cd4fceb5388e742a057aa3b1deeabad3cffff7606fb58f6c4000aa2493fd0

Findings



Critical	0 (0.00%)
Major	2 (11.76%)
Minor	4 (23.53%)
Informational	11 (64.71%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CFT-1	Lack of Input Validation	Volatile Code	Informational	Declined
CRI-1	Lack of Input Validation	Volatile Code	Informational	Declined
CRT-1	Lack of Input Validation	Volatile Code	Informational	Declined
IFB-1	Lack of Input Validation	Volatile Code	Informational	Declined
IFB-2	Proper Usage of `public` And `external` Type	Optimization	Informational	Declined
IFB-3	Missing Check Activity End Time	Logical Issue	Informational	Declined
IFB-4	Comparison Condition	Logical Issue	Minor	Resolved
IFB-5	Missing Emit Events	Optimization	Informational	Declined
IFB-6	Design of Function `finalWithdraw()`	Logical Issue	Minor	Resolved
IFB-7	Administrator Capability	Logical Issue	Major	Acknowledged
IFO-1	Lack of Input Validation	Volatile Code	Informational	Declined
IFO-2	Missing Emit Events	Optimization	Informational	Declined
IFO-3	Missing Check Activity End Time	Logical Issue	Informational	Declined
IFO-4	Administrator Capability	Logical Issue	Major	Acknowledged
IFO-5	Design of Function `finalWithdraw()`	Logical Issue	Minor	Resolved
IFO-6	Comparison Condition	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
IFO-7	Proper Usage of `public` And `external` Type	Optimization	Informational	Declined

CFT-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	swap-core-audit/contracts/CherryFactory.sol: 17~19, 42~45, 47~50	Declined

Description

The assigned value to `feeToSetter` in the constructor should be verified as a non-zero value to prevent being mistakenly assigned as `address(0)`.

The assigned value to `feeTo` in function `setFeeTo()` should be verified as non zero value to prevent being mistakenly assigned as `address(0)`.

The assigned value to `feeToSetter` in function `setFeeToSetter()` should be verified as non zero value to prevent being mistakenly assigned as `address(0)`.

Recommendation

Check that the address is not zero by adding following checks.

```
require(_feeToSetter != address(0), "_feeToSetter is a zero address");
```

Alleviation

No alleviation.

CRI-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	swap-periphery-audit/contracts/CherryRouter01.sol: 20~23	Declined

Description

The assigned value to `factory` and `WETH` should be verified as a non-zero value to prevent being mistakenly assigned as `address(0)`.

Recommendation

Check that the address is not zero by adding following checks.

```
require(_factory != address(0), "_factory is a zero address");
require(_WETH != address(0), "_WETH is a zero address");
```

Alleviation

No alleviation.

CRT-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	swap-periphery-audit/contracts/CherryRouter.sol: 24~27	Declined

Description

The assigned value to `factory` and `WETH` should be verified as a non-zero value to prevent being mistakenly assigned as `address(0)`.

Recommendation

Check that the address is not zero by adding following checks.

```
require(_factory != address(0), "_factory is a zero address");
require(_WETH != address(0), "_WETH is a zero address");
```

Alleviation

No alleviation.

IFB-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	swap-farm-audit/contract/IFOByProxy.sol: 55~64	Declined

Description

The assigned value to `lpToken`, `offeringToken`, and `adminAddress` should be verified as non zero value to prevent being mistakenly assigned as `address(0)`.

The assigned value to `endBlock` should be greater than `startBlock`.

The assigned value to `offeringAmount` and `raisingAmount` should be greater than zero.

The assigned value to `offeringAmount` should be greater than zero.

The assigned value to `raisingAmount` should be greater than zero.

Recommendation

Check that the address is not zero by adding following checks.

```
require(_lpToken != address(0), "_lpToken is a zero address");
require(offeringToken != address(0), "offeringToken is a zero address");
require(adminAddress != address(0), "adminAddress is a zero address");
```

Check that the assigned value to `endBlock` is greater than `startBlock`, like:

```
require(_startBlock < _endBlock, "_startBlock equals to or less than _endBlock");
```

Check that assigned value to both `offeringAmount` and `raisingAmount` are greater than zero, like:

```
require(offeringAmount > 0, "offeringAmount is zero");
require(raisingAmount > 0, "raisingAmount is zero");
```

Alleviation

No alleviation.

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

Category	Severity	Location	Status
Optimization	Informational	swap-farm-audit/contract/IFOByProxy.sol: 47, 71, 76, 81, 93, 144	Declined

Description

`public` functions that are never called by the contract could be declared `external`.

Examples:

`initialize()`, `setOfferingAmount()`, `setRaisingAmount()`, `deposit()`, `harvest()` and `finalWithdraw()`.

Alleviation

No alleviation.

IFB-3 | Missing Check Activity End Time

Category	Severity	Location	Status
Logical Issue	Informational	swap-farm-audit/contract/IFOByProxy.sol: 144	Declined

Description

`finalWithdraw()` ables to withdraw `offeringToken` by the admin, however, `offeringToken` should be distributed to users first.

Recommendation

Consider checking activity end time, like:

```
require (block.number  > endBlock + 7 days, 'waiting for a while');
```

Alleviation

No alleviation.

IFB-4 | Comparison Condition

Category	Severity	Location	Status
Logical Issue	Minor	swap-farm-audit/contract/IFOByProxy.sol: 145~146	Resolved

Description

Miss the equal case in the require checks.

Recommendation

Consider adding the equal case in the require checks, like:

```
function finalWithdraw(uint256 _lpAmount, uint256 _offerAmount) public onlyAdmin {
  ...
  require (_lpAmount <= lpToken.balanceOf(address(this)), 'not enough token 0');
  require (_offerAmount <= offeringToken.balanceOf(address(this)), 'not enough token 1');
  ...
}
```

Alleviation

The develop team heeded our advice and resolved this issue in commit 0e384392cf9917119dd57b2434344146d4af6dae.

IFB-5 | Missing Emit Events

Category	Severity	Location	Status
Optimization	Informational	swap-farm-audit/contract/IFOByProxy.sol: 47, 71, 76	Declined

Description

Description:

Several key actions are defined without event declarations.

Examples:

`initialize()`, `setOfferingAmount()` and `setRaisingAmount()`.

Recommendation

Consider emitting events for key actions.

Alleviation

No alleviation.

IFB-6 | Design of Function `finalWithdraw()`

Category	Severity	Location	Status
Logical Issue	Minor	swap-farm-audit/contract/IFOByProxy.sol: 147~152	Resolved

Description

There are `safeTransfer` for two tokens in a single function `finalWithdraw()`, failure on transferring of one token has a side effect on the other transferring.

Recommendation

Avoid `safeTransfer` two tokens in a single function.

Alleviation

The development team heeded our advice and resolved this issue in commit 2911d9021b5f8afddb92ba295e68d016684363fe.

IFB-7 | Administrator Capability

Category	Severity	Location	Status
Logical Issue	Major	swap-farm-audit/contract/IFOByProxy.sol: 148	Acknowledged

Description

To bridge the trust gap between administrator and users, administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrator can transfer offering tokens and raising tokens to own account under unpredicted cases via `finalWithdraw` function.

Example:

```
function finalWithdraw(uint256 _lpAmount, uint256 _offerAmount) public onlyAdmin {
  require (_lpAmount < lpToken.balanceOf(address(this)), 'not enough token 0');
  require (_offerAmount < offeringToken.balanceOf(address(this)), 'not enough token 1');
  lpToken.safeTransfer(address(msg.sender), _lpAmount);
  offeringToken.safeTransfer(address(msg.sender), _offerAmount);
}
```

Recommendation

The advantage of `finalWithdraw` function in the protocol is that the administrator reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the downside of `finalWithdraw` function, where the treasury in this contract can be migrated to administrator's addresses.

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients. Any plan to call this `finalWithdraw` methods is better to move to the execution queue of Timelock, and also emit events.

Alleviation

The development team heeded part of our advice and limited the admin to execute the function `finalWithdraw` only after the end of the activity, in the commit 4bd51fa20e83a625e52ecded3d3cb8bccb97e308.

IFO-1 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	swap-farm-audit/contract/IFO.sol: 51~59	Declined

Description

The assigned value to `lpToken`, `offeringToken`, and `adminAddress` should be verified as non zero value to prevent being mistakenly assigned as `address(0)`.

The assigned value to `endBlock` should be greater than `startBlock`.

The assigned value to `offeringAmount` and `raisingAmount` should be greater than zero.

The assigned value to `offeringAmount` should be greater than zero.

The assigned value to `raisingAmount` should be greater than zero.

Recommendation

Check that the address is not zero by adding following checks.

```
require(_lpToken != address(0), "_lpToken is a zero address");
require(offeringToken != address(0), "offeringToken is a zero address");
require(adminAddress != address(0), "adminAddress is a zero address");
```

Check that the assigned value to `endBlock` is greater than `startBlock`, like:

```
require(_startBlock < _endBlock, "_startBlock equals to or less than _endBlock");
```

Check that assigned value to both `offeringAmount` and `raisingAmount` are greater than zero, like:

```
require(offeringAmount > 0, "offeringAmount is zero");
require(raisingAmount > 0, "raisingAmount is zero");
```

Alleviation

No alleviation.

IFO-2 | Missing Emit Events

Category	Severity	Location	Status
Optimization	Informational	swap-farm-audit/contract/IFO.sol: 67, 72	Declined

Description

Description:

Several key actions are defined without event declarations.

Examples:

`setOfferingAmount()` and `setRaisingAmount()`.

`initialize()`, `setOfferingAmount()` and `setRaisingAmount()`.

Recommendation

Consider emitting events for key actions.

Alleviation

No alleviation.

IFO-3 | Missing Check Activity End Time

Category	Severity	Location	Status
Logical Issue	Informational	swap-farm-audit/contract/IFO.sol: 138	Declined

Description

`finalWithdraw()` ables to withdraw `offeringToken` by the admin, however, `offeringToken` should be distributed to users first.

Recommendation

Consider checking activity end time, like:

```
require (block.number  > endBlock + 7 days, 'waiting for a while');
```

Alleviation

No alleviation.

IFO-4 | Administrator Capability

Category	Severity	Location	Status
Logical Issue	Major	swap-farm-audit/contract/IFO.sol: 140	Acknowledged

Description

To bridge the trust gap between administrator and users, administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following capability of the administrator:

- Administrator can transfer offering tokens and raising tokens to own account under unpredicted cases via `finalWithdraw` function.

Example:

```
function finalWithdraw(uint256 _lpAmount, uint256 _offerAmount) public onlyAdmin {
  require (_lpAmount < lpToken.balanceOf(address(this)), 'not enough token 0');
  require (_offerAmount < offeringToken.balanceOf(address(this)), 'not enough token 1');
  lpToken.safeTransfer(address(msg.sender), _lpAmount);
  offeringToken.safeTransfer(address(msg.sender), _offerAmount);
}
```

Recommendation

The advantage of `finalWithdraw` function in the protocol is that the administrator reserves the ability to rescue the assets in this contract under unexpected cases. It is also worthy of note the downside of `finalWithdraw` function, where the treasury in this contract can be migrated to administrator's addresses.

To improve the trustworthiness of the project, any dynamic runtime changes on the protocol should be notified to clients. Any plan to call this 'finalWithdraw' methods is better to move to the execution queue of Timelock, and also emit events.

Alleviation

The development team heeded part of our advice and limited the admin to execute the function `finalWithdraw` only after the end of the activity, in the commit c7c7cc1442ffd77b89395cbc6fae168feff1172b.

IFO-5 | Design of Function `finalWithdraw()`

Category	Severity	Location	Status
Logical Issue	Minor	swap-farm-audit/contract/IFO.sol: 141~142	Resolved

Description

There are `safeTransfer` for two tokens in a single function `finalWithdraw()`, failure on transferring of one token has a side effect on the other transferring.

Recommendation

Avoid `safeTransfer` two tokens in a single function.

Alleviation

The development team heeded our advice and resolved this issue in commit 66979307cf84f2754c6c1099e69f5d3f4ac1541f.

IFO-6 | Comparison Condition

Category	Severity	Location	Status
Logical Issue	Minor	swap-farm-audit/contract/IFO.sol: 139~140	Resolved

Description

Miss the equal case in the require checks.

Recommendation

Consider adding the equal case in the require checks, like:

```
function finalWithdraw(uint256 _lpAmount, uint256 _offerAmount) public onlyAdmin {
  ...
  require (_lpAmount <= lpToken.balanceOf(address(this)), 'not enough token 0');
  require (_offerAmount <= offeringToken.balanceOf(address(this)), 'not enough token 1');
  ...
}
```

Alleviation

The develop team heeded our advice and resolved this issue in commit 0e384392cf9917119dd57b2434344146d4af6dae.

IFO-7 | Proper Usage of `public` And `external` Type

Category	Severity	Location	Status
Optimization	Informational	swap-farm-audit/contract/IFO.sol: 67, 72, 77, 89, 138	Declined

Description

`public` functions that are never called by the contract could be declared `external`.

Examples:

`setOfferingAmount()`, `setRaisingAmount()`, `deposit()`, `harvest()` and `finalWithdraw()`.

Alleviation

No alleviation.

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.