Security Assessment

CoPuppy II

Jul 30th, 2021

Table of Contents

Summary

This report has been prepared for CoPuppy to discover issues and vulnerabilities in the source code of the CoPuppy II project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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 that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	CoPuppy II
Platform	BSC
Language	Solidity
Codebase	https://github.com/copuppy/CP
Commit	266ab31bf55e5e1f7155bbc11adbcc621208fce4

Audit Summary

Delivery Date	Jul 30, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	4	0	0	2	2	0
Informational	5	0	0	5	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
CLP	LPFarm/CakeLPFarm.sol	9705f3a30eb5dc235b05e98ab2452723539badd4e42b574256291c2739bb5078
UPL	LPFarm/UpgradeProxy.sol	9827cd4f66dd1f6aff62ea4dfc5f8ca3aac2e4582cc1ce24d786e43447d3b772

Understandings

Overview

CoPuppy is an upgradeable mining project. When users deposit LP tokens, these tokens will be automatically deposited into Pancake Pool to earn cake rewards. Users can get `ckey` tokens as rewards. The owner can distribute cake rewards through the function `swapAndLiquify()`. 40% of the rewards will be distributed to the owner directly, and the rest will be converted into LP tokens and deposit into Pancake Pool. After that, the pool's exchange rate between Pancake Pool and CoPuppy Pool will increase. So users can get extra LP tokens.

Privileged Functions

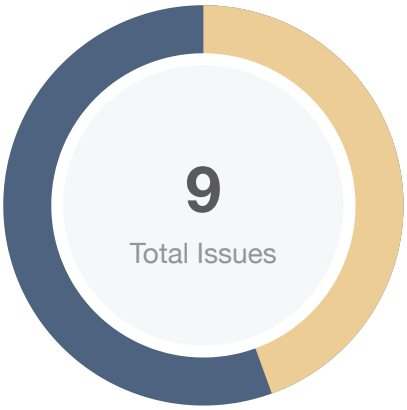
The contract `CakeLPFarm` contains the following privileged functions that are restricted by the `onlyOwner` modifier. They are used to modify the contract configurations and address attributes. We grouped these functions below:

- function add(uint256 _allocPoint, IERC20 _lpToken, bool _withUpdate)
- function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate)
- function setPidMap(uint256 _pid, uint256 _dPid)
- function setPerBlcokReward(uint256 reward)
- function depositInterest(uint256 _pid)
- function swapAndLiquifyAll()
- function swapAndLiquify(uint256 _pid, address[] memory routerPath)

The contract `UpgradeProxy` contains the following privileged functions that are restricted by the `ifAdmin` modifier. They are used to modify the contract configurations and address attributes. We grouped these functions below:

- function changeAdmin(address newAdmin)
- function upgradeTo(address newImplementation)
- function upgradeToAndCall(address newImplementation, bytes calldata data)

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**4** (44.44%)
Informational	**5** (55.56%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CLP-01	Centralization Risk	**Centralization / Privilege**	**Minor**	**Acknowledged**
CLP-02	`add()` Function Not Restricted	Logical Issue	Minor	Resolved
CLP-03	Strengthen Transfer Security	Logical Issue	Minor	Resolved
CLP-04	Missing Emit Events	Coding Style	Informational	Resolved
CLP-05	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Resolved
CLP-06	Hardcode Address	Logical Issue	Informational	Resolved
CLP-07	Uninitialized variable `bonusEndBlock`	Logical Issue	Informational	Resolved
CLP-08	Function Name Typo	Coding Style	Informational	Resolved
UPL-01	Centralization Risk	**Centralization / Privilege**	**Minor**	**Acknowledged**

CLP-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Minor	LPFarm/CakeLPFarm.sol: 937	Acknowledged

Description

In the contract `CakeLPFarm`, the role `owner` has the authority over the following function:

- function add(uint256 _allocPoint, IERC20 _lpToken, bool _withUpdate)
- function set(uint256 _pid, uint256 _allocPoint, bool _withUpdate)
- function setPidMap(uint256 _pid, uint256 _dPid)
- function setPerBlcokReward(uint256 reward)
- function depositInterest(uint256 _pid)
- function swapAndLiquifyAll()
- function swapAndLiquify(uint256 _pid, address[] memory routerPath)

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets. Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The development team responded that after the contract is deployed, they will transfer the `owner` to the timelock contract, which will be locked for 48 hours.

CLP-02 | `add()` Function Not Restricted

Category	Severity	Location	Status
Logical Issue	Minor	LPFarm/CakeLPFarm.sol: 937	Resolved

Description

When the same LP token is added into a pool more than once in function `add()`, the total amount of reward in function `updatePool()` will be incorrectly calculated. The current implementation is relying on the operation correctness to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using a mapping of `addresses` -> `booleans`, which can restricted the same address being added twice.

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-03 | Strengthen Transfer Security

Category	Severity	Location	Status
Logical Issue	Minor	LPFarm/CakeLPFarm.sol: 1031, 1059, 1151	Resolved

Description

There are a lot of transfer operations in functions `deposit()`, `withdraw()` and `swapAndLiquify()`, add a reentrant would be safer.

Recommendation

Consider adding a modifier as below:

```
bool private _status;
modifier nonReentrant() {
    require(!_status, 'reentrant call');
    _status = true;
    _;
   _status = false;
}
```

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	LPFarm/CakeLPFarm.sol: 963, 967	Resolved

Description

Function that affect the status of sensitive variables should be able to emit events as notifications to customers:

- `setPidMap()`
- `setPerBlcokReward()`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-05 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	LPFarm/CakeLPFarm.sol: 954, 963, 983, 1003, 1031, 1059, 1096, 1108, 1113, 1151	Resolved

Description

There's no sanity check to validate if a pool is existing.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `set()`, `setPidMap()`, `deposit()`, `withdraw()`, `depositInterest()`, 'calculateFee()', `getUserAmount()`, `swapAndLiquify()`, `pendingSushi()` and `updatePool()`.

```
1 modifier validatePoolByPid(uint256 _pid) {
2     require (_pid < poolInfo . length , "Pool does not exist") ;
3     _;
4 }
```

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-06 | Hardcode Address

Category	Severity	Location	Status
Logical Issue	Informational	LPFarm/CakeLPFarm.sol: 914, 917	Resolved

Description

1. if `msg.sender` is zero address, nobody can call the function `initialize()`.
2. The ERC20 token `ckey` is a zero address.
3. For other hardcode addresses, please double-check.

Recommendation

We recommended changing to the correct address.

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-07 | Uninitialized variable `bonusEndBlock`

Category	Severity	Location	Status
Logical Issue	Informational	LPFarm/CakeLPFarm.sol: 895	Resolved

Description

The variable `bonusEndBlock` is uninitialized, but it's used in function `getMultiplier()`. Is it necessary to initialize in function `initialize()`?

Alleviation

The development team removed this variable `bonusEndBlock` in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

CLP-08 | Function Name Typo

Category	Severity	Location	Status
Coding Style	Informational	LPFarm/CakeLPFarm.sol: 967~968	Resolved

Description

In the following code snippet, `setPerBlcokReward` should be `setPerBlockReward`.

```
1     function setPerBlcokReward(uint256 reward) public onlyOwner {
2         sushiPerBlock = reward;
3     }
```

Recommendation

We recommend correcting all typos in the contract.

Alleviation

The development team resolved this issue in commit `066e9e7465769b86c674eae1c93671cc05a30aee`.

UPL-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	LPFarm/UpgradeProxy.sol: 456, 470, 481	**Acknowledged**

Description

In the contract `TransparentUpgradeableProxy`, the role `admin` has the authority over the following function:

- function changeAdmin(address newAdmin)
- function upgradeTo(address newImplementation)
- function upgradeToAndCall(address newImplementation, bytes calldata data)

Any compromise to the `admin` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `admin` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets. Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The development team responded that after the contract is deployed, they will transfer the `admin` to the timelock contract, which will be locked for 48 hours.

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.

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.

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.