



Security Assessment

Dog Moon

May 24th, 2021



Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

CTC-01 : Lack of Input Validation

CTC-02 : Incorrect error message

CTC-03 : Redundant code

CTC-04 : Centralized risk in `addLiquidity`

CTC-05 : Return value not handled

CTC-06 : 3rd party dependencies

CTC-07 : Missing event emitting

CTC-08 : Privileged ownership

CTC-09 : Typos in the contract

CTC-10 : The purpose of function `deliver`

CTC-11 : Possible to gain ownership after renouncing the contract ownership

Formal Verification Requests

Appendix

Disclaimer

About

Summary

This report has been prepared for Dog Moon 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 Manual Review and Static Analysis 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	Dog Moon
Platform	Heco Chain
Language	Solidity
Codebase	https://hecoinfo.com/address/0x67bab912ee30074cf9a94826e2e02d9936842781#code
Commits	Address of deployed contract: 0x67bab912ee30074cf9a94826e2e02d9936842781

Audit Summary

Delivery Date	May 24, 2021
Audit Methodology	Manual Review, Static Analysis
Key Components	

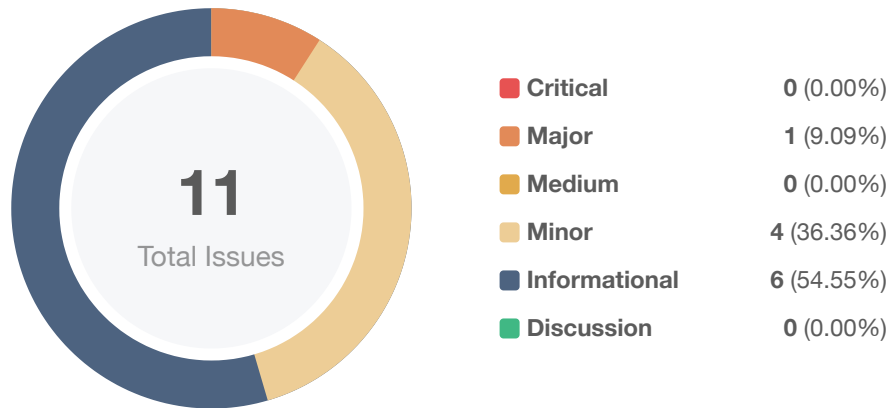
Vulnerability Summary

Total Issues	11
● Critical	0
● Major	1
● Medium	0
● Minor	4
● Informational	6
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CTC	CoinToken.sol	9505e1ec73f855bd1f18c0c6484625eddbbc56723c3f786dabdf0236b3702bfc

Findings



ID	Title	Category	Severity	Status
CTC-01	Lack of Input Validation	Volatile Code	● Informational	ⓘ Acknowledged
CTC-02	Incorrect error message	Logical Issue	● Minor	ⓘ Acknowledged
CTC-03	Redundant code	Logical Issue	● Informational	ⓘ Acknowledged
CTC-04	Centralized risk in <code>addLiquidity</code>	Centralization / Privilege	● Major	☑ Resolved
CTC-05	Return value not handled	Volatile Code	● Informational	ⓘ Acknowledged
CTC-06	3rd party dependencies	Control Flow	● Minor	ⓘ Acknowledged
CTC-07	Missing event emitting	Coding Style	● Informational	ⓘ Acknowledged
CTC-08	Privileged ownership	Centralization / Privilege	● Minor	ⓘ Acknowledged
CTC-09	Typos in the contract	Coding Style	● Informational	ⓘ Acknowledged
CTC-10	The purpose of function <code>deliver</code>	Control Flow	● Informational	ⓘ Acknowledged
CTC-11	Possible to gain ownership after renouncing the contract ownership	Logical Issue, Centralization / Privilege	● Minor	☑ Resolved

CTC-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	CoinToken.sol: 742~744	ⓘ Acknowledged

Description

The input parameters `routerAddress` and `tokenOwner` of the constructor of contract `CoinToken` should be verified as a non-zero address.

Alleviation

No alleviation.

CTC-02 | Incorrect error message

Category	Severity	Location	Status
Logical Issue	● Minor	CoinToken.sol: 857	ⓘ Acknowledged

Description

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

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded" .

Alleviation

No alleviation.

CTC-03 | Redundant code

Category	Severity	Location	Status
Logical Issue	● Informational	CoinToken.sol: 1118	ⓘ Acknowledged

Description

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

Recommendation

The following code can be removed:

```
1 ... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {  
2     _transferStandard(sender, recipient, amount);  
3 } ...
```

Alleviation

No alleviation.

CTC-04 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	● Major	CoinToken.sol: 1093	🟢 Resolved

Description

```
1 // add the liquidity
2 uniswapV2Router.addLiquidityETH{value: ethAmount}(
3     address(this),
4     tokenAmount,
5     0, // slippage is unavoidable
6     0, // slippage is unavoidable
7     owner(),
8     block.timestamp
9 );
```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `DogMoon-HT` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent 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 confirmed that the owner is a black hole and all the liquidity are locked to the black hole.

CTC-05 | Return value not handled

Category	Severity	Location	Status
Volatile Code	● Informational	CoinToken.sol: 1098	ⓘ Acknowledged

Description

The return values of function `addLiquidityETH` are not properly handled.

```
1     uniswapV2Router.addLiquidityETH(value: ethAmount){
2         address(this),
3         tokenAmount,
4         0, // slippage is unavoidable
5         0, // slippage is unavoidable
6         owner(),
7         block.timestamp
8     );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

No alleviation.

CTC-06 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	● Minor	CoinToken.sol: 744~747	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of the DogMoon protocol requires the interaction 3rd party protocol for adding liquidity to DogMoon-HT pool and swap tokens. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

No alleviation.

CTC-07 | Missing event emitting

Category	Severity	Location	Status
Coding Style	● Informational	CoinToken.sol: 677	ⓘ Acknowledged

Description

In contract `DogMoon`, there are a bunch of functions can change state variables. However, these function do not emit event to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that are possible to be changed during runtime.

Alleviation

No alleviation.

CTC-08 | Privileged ownership

Category	Severity	Location	Status
Centralization / Privilege	● Minor	CoinToken.sol: 447~460	ⓘ Acknowledged

Description

The owner of contract `DogMoon` has the permission to:

1. change the address that can receive LP tokens,
2. lock the contract,
3. exclude/include addresses from rewards/fees,
4. set `taxFee`, `liquidityFee` and `_maxTxAmount`,
5. enable `swapAndLiquifyEnabled`

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

No alleviation.

CTC-09 | Typos in the contract

Category	Severity	Location	Status
Coding Style	● Informational	CoinToken.sol: 719	ⓘ Acknowledged

Description

There are several typos in the code and comments.

1. In the following code snippet, `tokensIntoLiquidity` should be `tokensIntoLiquidity`.

```
1 event SwapAndLiquify(  
2     uint256 tokensSwapped,  
3     uint256 ethReceived,  
4     uint256 tokensIntoLiquidity  
5 );
```

2. `recieve` should be `receive` and `swaping` should be `swapping` in the line of comment `//to recieve ETH from uniswapV2Router when swaping`.

Recommendation

We recommend correcting all typos in the contract.

Alleviation

No alleviation.

CTC-10 | The purpose of function `deliver`

Category	Severity	Location	Status
Control Flow	● Informational	CoinToken.sol: 820~827	ⓘ Acknowledged

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the DogMoon token balance of the caller by `rAmount`, which is `tAmount` reduces the transaction fee. Then, the function adds `tAmount` to variable `_tFeeTotal`, which represents the contract's total transaction fee. We wish the team could explain more on the purpose of having such functionality.

Alleviation

No alleviation.

CTC-11 | Possible to gain ownership after renouncing the contract ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	● Minor	CoinToken.sol: 447~460	☑ 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>

Alleviation

The owner is transferred to the black hole.

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.

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.

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.

