CERTIK

Security Assessment

City of Dream

Apr 18th, 2022

Table of Contents

Summary

This report has been prepared for City of Dream to discover issues and vulnerabilities in the source code of the City of Dream project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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;
- 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	City of Dream
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x73Eb6947D72ED1979e6A935F516212A7f593bC44 https://github.com/COD-Contract/COD/tree/main
Commit	4db4c34528923681d93f5c3c35594baa1d497bbe

Audit Summary

Delivery Date	Apr 18, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

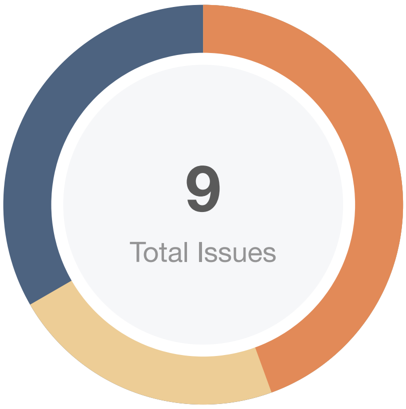
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	4	0	0	4	0	0	0
Medium	0	0	0	0	0	0	0
Minor	2	0	0	1	0	0	1
Informational	3	0	0	0	0	0	3
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	Repo	File	SHA256 Checksum
COD	mainnet	COD.sol	8a5f7b9d88cc743e5dcefc7c5970455486df9dab5aae2091557de66522740ee2

Findings



Critical	**0** (0.00%)
Major	**4** (44.44%)
Medium	**0** (0.00%)
Minor	**2** (22.22%)
Informational	**3** (33.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
COD-01	Centralization Risks in COD.sol	**Centralization / Privilege**	**Major**	Acknowledged
COD-02	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
COD-03	Potential Loss of Tokens	Logical Issue	Major	Acknowledged
COD-04	Centralization risk in `_pool`	**Centralization / Privilege**	**Major**	Acknowledged
COD-05	Missing Zero Address Validation	Volatile Code	Minor	Resolved
COD-06	Third Party Dependencies	Volatile Code	Minor	Acknowledged
COD-07	Improper Usage of `public` and `external` Type	Gas Optimization	Informational	Resolved
COD-08	Unlocked Compiler Version	Language Specific	Informational	Resolved
COD-09	Useless Function	Coding Style	Informational	Resolved

COD-01 | Centralization Risks In COD.sol

Category	Severity	Location	Status
Centralization / Privilege	● Major	COD.sol: 326, 334, 511, 517, 521, 526, 530, 534, 538, 543	ⓘ Acknowledged

Description

In the contract `COD` the role `_owner` has authority over the functions listed below.

- `setMarket`: manage the state variable `_markets`
- `setPay`: manage the state variable `_pay`
- `setPool`: manage the state variable `_pool`
- `setBind`: manage the state variable `_bind`
- `setWhitelistFrom`: manage the state variable `_whitelistFrom`
- `setWhitelistTo`: manage the state variable `_whitelistTo`
- `setBidirectionWhitelist`: manage the state variables `_whitelistFrom` and `_whitelistTo`
- `setBatchBidirectionWhitelist` manage the state variables `_whitelistFrom` and `_whitelistTo`

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority, and change the configuration of this contract.

In the contract `Ownable` the role `_owner` has authority over the functions listed below.

- `renounceOwnership`: function call `_transferOwnership` and manage the state variable `_owner`
- `transferOwnership`: function call `_transferOwnership` and manage the state variable `_owner`

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

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[COD]: The team acknowledged this issue and decided not to change the codebase this time due to some necessary logic need to retain ownership.

COD-02 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	COD.sol: 508	ⓘ Acknowledged

Description

All of the COD tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute COD tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

[COD]: The COD tokens had been distributed after the contract release.

COD-03 | Potential Loss Of Tokens

Category	Severity	Location	Status
Logical Issue	Major	COD.sol: 651~654	Acknowledged

Description

We do not know the internal implementation of the function `IBind(_bind).getReferrer(receiver)`. If the function `IBind(_bind).getReferrer(receiver)` returns a zero address, the token sent to the referrer will be lost.

Suppose that the addresses mentioned below are not in the whitelist, there are several possible cases where the COD tokens are lost when transferring tokens.

- If a contract sends the COD tokens to an address that is not in a binding relationship, the referrer of the `recipient` will be set to the zero address in line 679. Then in line 652, the variable `referrer` is set to a zero address. As a result, the COD tokens sent to the variable `referrer` will be lost.
- According to the implementation of the function `__bind`, the contract does not have a referrer. if a contract sends the COD tokens to another contract, the variable `referrer` will be set to the zero address in line 652, causing the same consequences as in the case above.

Recommendation

We recommend checking whether the referrer is a zero address before transferring tokens. If the referrer is indeed a zero address, the logic related to referral fees needs to be adjusted as well.

Alleviation

[COD]: The team will adjust the 0x0 address to the vertex number when binding in the contract `Bind`.

COD-04 | Centralization Risk In `_pool`

Category	Severity	Location	Status
Centralization / Privilege	Major	COD.sol: 649	Acknowledged

Description

Over time, the code snippet below will lead that the `_pool` address will accumulate a significant amount of tokens.

```
_balances[_pool] = _balances[_pool] + bonusFee;
```

Recommendation

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

[COD]: The role of the `_pool` is to provide a temporary storage address for the prize of the upcoming Global Lottery Pool.

COD-05 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	COD.sol: 518, 522, 527	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

File: COD.sol (Line 518, Function `COD.setPay`)

```
_pay = addr;
```

- `addr` is not zero-checked before being used.

File: COD.sol (Line 522, Function `COD.setPool`)

```
_pool = addr;
```

- `addr` is not zero-checked before being used.

File: COD.sol (Line 527, Function `COD.setBind`)

```
_bind = bind;
```

- `bind` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[COD]: The team resolved this issue by adding a zero-check for the passed-in address value.

COD-06 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	COD.sol: 475	Acknowledged

Description

The contract is serving as the underlying entity to interact with third party `IBind` protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of binding relationships requires interaction with `IBind`. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[COD]: The team will continuously monitor the Bind contract to ensure that no problems occur.

COD-07 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	COD.sol: 326, 334, 511, 517, 521, 526, 530, 534, 538, 543, 550, 554, 558, 563, 573, 579, 584, 590, 607, 613	Resolved

Description

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

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

[COD]: The team resolved this issue by declaring the aforementioned `public` functions as `external`.

COD-08 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	COD.sol: 467	Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to different compiler versions. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.0` the contract should contain the following line:

```
pragma solidity 0.8.0;
```

Alleviation

[COD]: The team resolved this issue by locking the compiler version to `0.8.0`.

COD-09 | Useless Function

Category	Severity	Location	Status
Coding Style	Informational	COD.sol: 699	Resolved

Description

The internal function `_burn` is not used.

File: COD.sol (Line 699, Contract `COD`)

```
function _burn(address account, uint256 amount) internal virtual {
```

Recommendation

We recommend removing this unused function.

Alleviation

[COD]: The team removed the aforementioned useless function `_burn`.

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.

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.

Volatile Code

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

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 you (“Customer” or 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, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.

CERTIK