CERTIK

Security Assessment

CyberStop

May 6th, 2021

Summary

This report has been prepared for CyberStop 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.

Addtionally, this audit is based on a premise that all external smart contracts were implemented safely. This protocol can be used more safely only if there is already a sufficient amount of CERT in `CertInitialOffering` contracts.

The security assessment resulted in 7 findings that ranged from minor 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	CyberStop
Description	CyberStop is a community platform for NFT products.You can create, sell, buy, collect and discover the most interesting NFT products from all over the world. CERT is a deflationary currency that will be issued by CyberStop.
Platform	BSC, HECO
Language	Solidity
Codebase	https://github.com/CyberStopExchange
Commits	1. 668967d5f0d9edca8de99ef70ac124d7d01bd381 2. 8d61ac7a6b642c57d456991fb061690b570b1fa2

Audit Summary

Delivery Date	May 06, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Total Issues	7
● Critical	0
● Major	0
● Medium	0
● Minor	2
● Informational	5
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CIO	CertInitialOffering.sol	b43bd62cbf6247aecbc33d0cb84b4e5e0b2e6d0234f9b6e5983c1c11222a8f1e
CIC	CertInitialOffering_bsc.sol	10ca3af72c4fa39729814b3a561a3fb310a611c1fb2ded106600fe3d4b050ef6
CIK	CertInitialOffering_heco.sol	c9c9ec5b9d76f21e5ef8e1a26aed2a90a09e8b35a533db53e07adcad0af48893
CTC	CertToken.sol	737e2c7379b0ae6ae5edfe7041b79a6ec50c078f3bf03913f3d6afca42090f9d

Certralization Roles

The CyberStop smart contract introduces an authorization.

Owner:

[CertInitialOffering.sol] [CertInitialOffering_bsc.sol] [CertInitialOffering_heco.sol]

- withdrawProvidedHUSD()

After reaching the end time defined in the contract, the owner of the contract can withdraw all `HUSD` of the contract.

- withdrawUnclaimedCERT()

30 days after the end time defined in the contract, regardless of whether users had claimed their own `CERT`, the contract owner can withdraw all `CERT` of the contract.

- burnCERT()

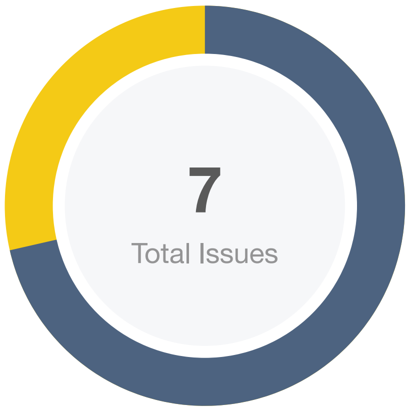
`CERT` that users did not purchase will be burned.

[CertToken.sol]

- setFee()

The state variable of `fee` is defined in the contract, which can affect the amount of deflationary currency.

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**0** (0.00%)
🟧 **Medium**	**0** (0.00%)
🟨 **Minor**	**2** (28.57%)
🟦 **Informational**	**5** (71.43%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
CIC-01	Discussion About the Reason For Burning the Remaining CERT	Logical Issue	● Informational	ⓘ Acknowledged
CIC-02	Discussion About The Source Of CERT Token	Logical Issue	● Minor	ⓘ Acknowledged
CIK-01	Discussion About the Reason For Burning the Remaining CERT	Logical Issue	● Informational	ⓘ Acknowledged
CIK-02	Discussion About The Source Of CERT Token	Logical Issue	● Minor	ⓘ Acknowledged
CIO-01	Naming Conventions	Coding Style	● Informational	✓ Resolved
CIO-02	Integer Overflow Risk	Mathematical Operations	● Informational	✓ Resolved
CIO-03	Zero Address in CertInitialOffering.sol	Logical Issue	● Informational	✓ Resolved

CIC-01 | Discussion About the Reason For Burning the Remaining CERT

Category	Severity	Location	Status
Logical Issue	● Informational	CertInitialOffering_bsc.sol: 598	ⓘ Acknowledged

Description

After user complete the purchase, if there are remaining CERT tokens, why should the remaining CERT tokens be burned?

Alleviation

Customer team response:
According to the distribution of tokens, if there are any remaining tokens, we won't let them return to the team. Deflation will make the CERT held by users more valuable.

CIC-02 | Discussion About The Source Of CERT Token

Category	Severity	Location	Status
Logical Issue	● Minor	CertInitialOffering_bsc.sol	ⓘ Acknowledged

Description

If the `CERT` tokens of these contracts are not enough. Some users may not claim equivalent amounts of `CERT` from them. When will these `CERT` tokens be transferred to these contracts? Will these tokens be transferred to these contracts before starting this activity?

Recommendation

Please make sure these contracts have enough `CERT` and users could claim equivalent amounts of `CERT`. For example, after users send `HUSD` to these contracts to buy `CERT`, then transfer equivalent `CERT` to them directly and add timelocks to the `CERT` balance.

Alleviation

Customer team response:
Required amount of CERT tokens will be manually transferred to the smart contract immediately after deployment.

CIK-01 | Discussion About the Reason For Burning the Remaining CERT

Category	Severity	Location	Status
Logical Issue	● Informational	CertInitialOffering_heco.sol: 598	ⓘ Acknowledged

Description

After user complete the purchase, if there are remaining CERT tokens, why should the remaining CERT tokens be burned?

Alleviation

Customer team response:
According to the distribution of tokens, if there are any remaining tokens, we won't let them return to the team. Deflation will make the CERT held by users more valuable.

CIK-02 | Discussion About The Source Of CERT Token

Category	Severity	Location	Status
Logical Issue	● Minor	CertInitialOffering_heco.sol	ⓘ Acknowledged

Description

If the `CERT` tokens of these contracts are not enough. Some users may not claim equivalent amounts of `CERT` from them. When will these `CERT` tokens be transferred to these contracts? Will these tokens be transferred to these contracts before starting this activity?

Recommendation

Please make sure these contracts have enough `CERT` and users could claim equivalent amounts of `CERT`. For example, after users send `HUSD` to these contracts to buy `CERT`, then transfer equivalent `CERT` to them directly and add timelocks to the `CERT` balance.

Alleviation

Customer team response:
Required amount of CERT tokens will be manually transferred to the smart contract immediately after deployment.

CIO-01 | Naming Conventions

Category	Severity	Location	Status
Coding Style	● Informational	CertInitialOffering.sol: 516~517(CertInitialOffering), 552(CertInitialOffering)	✓ Resolved

Description

Parameters `START`, `END`, `_husdAmt` are not in mixedCase.

Recommendation

Function arguments,local and state variable names should use mixedCase.Examples: `start`,`end`,`husdAmt`.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 8d61ac7a6b642c57d456991fb061690b570b1fa2.

CIO-02 | Integer Overflow Risk

Category	Severity	Location	Status
Mathematical Operations	● Informational	CertInitialOffering.sol: 556~558(CertInitialOffering)	✓ Resolved

Description

Using `+` in the method directly to calculate the value of the variable may overflow. `SafeMath` provides a method to verify overflow, and it is safer to use the method provided.

Recommendation

Using the `add()` function in `SafeMath` library for mathematical operations.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 8d61ac7a6b642c57d456991fb061690b570b1fa2.

CIO-03 | Zero Address in CertInitialOffering.sol

Category	Severity	Location	Status
Logical Issue	● Informational	CertInitialOffering.sol: 1(CertInitialOffering)	Resolved

Description

The assigned value to `CERT`,`HUSD` should be verified as non zero value to prevent being mistakenly assigned as `address(0)` in constructor of contract.

Recommendation

Check that the address is not zero by adding following checks in the constructor of contract `CertInitialOffering.sol`.Example:

```
constructor(address cert, address husd) public {
        require(cert!=address(0),"cert is a zero address!");
        require(husd!=address(0),"husd is a zero address!");
        ...
    }
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 8d61ac7a6b642c57d456991fb061690b570b1fa2.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

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 but rather 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.

CERTIK