Security Assessment

UINC

May 29th, 2021

Table of Contents

Summary

This report has been prepared for UINC 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 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	UINC
Platform	Heco
Language	Solidity
Codebase	https://github.com/UintCola/Heco20-Token
Commits	2964e7d7ca940feff6ad9736279b6e4144fe0a98

Audit Summary

Delivery Date	May 29, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

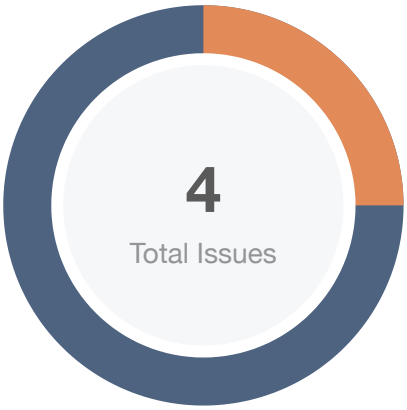
Vulnerability Summary

Total Issues	4
Critical	0
Major	1
Medium	0
Minor	0
Informational	3
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
UIN	UINC.sol	dd9608ef04f1ab17b68700abcc7856adfdbcdb9088e5b7eda3f4ff9f6b5fc308

Findings



Critical	**0** (0.00%)
Major	**1** (25.00%)
Medium	**0** (0.00%)
Minor	**0** (0.00%)
Informational	**3** (75.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
UIN-01	Proper Usage of “public” and “external” type	Coding Style	Informational	Acknowledged
UIN-02	Too Many Digits	Coding Style	Informational	Resolved
UIN-03	Usage of input parameter	Logical Issue	Informational	Acknowledged
UIN-04	Centralized Risks	**Centralization / Privilege**	**Major**	**Acknowledged**

UIN-01 | Proper Usage of “public” and “external” type

Category	Severity	Location	Status
Coding Style	Informational	UINC.sol: 1307(HecoUINCToken), 1314(HecoUINCToken), 1319(HecoUINCToken), 1331(HecoUINCToken), 1359(UINCToken), 1367(UINCToken), 1372(UINCToken), 1385(UINCToken)	Acknowledged

Description

“public” functions that are never called by the contract should be declared “external”. For example if inputs are arrays, “external” functions are more efficient than “public” functions.

Examples: Functions like : `setLockAddr()`, `getLockAddr()`, `depositTo()`, `withdrawTo()`, `mint()`, `getMinter()`, `addMinter()`, `delMinter()`

Recommendation

Consider using the “external” attribute for functions never called from the contract.

UIN-02 | Too Many Digits

Category	Severity	Location	Status
Coding Style	Informational	UINC.sol: 1348(UINCToken), 1349(UINCToken)	Resolved

Description

Literals with many digits are difficult to read and review.

```
1348  uint256 private constant preMineSupply = 50000000 * 1e18;
1349  uint256 private constant maxSupply = 1000000000 * 1e18;
```

Recommendation

Consider using scientific notation to improve readability.

Alleviation

The team heeded our advice and resolved this issue in commit 1bed3f4f7217c9869bbe21eb0acd900efd8907da

UIN-03 | Usage of input parameter

Category	Severity	Location	Status
Logical Issue	Informational	UINC.sol: 1319(HecoUINCToken)	Acknowledged

Description

According to the codes, the usage of the input parameter `UINCAddr` in method `depositTo()` is not quite clear. Can you explain the reason for the length check (`UINCAddr.length <= 64`)?

Recommendation

Need a discussion.

Alleviation

The development team replied that this parameter is used to aviod mistake putting using wrongl `depositTo()` Tx hash address.

UIN-04 | Centralized Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	UINC.sol: 1331	**Acknowledged**

Description

To bridge the trust gap between the administrator and users, the 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:

- User tokens could be lock by calling function `depositTo()` in address `_lock_UINC_addr`
- Admin can transfer the locked tokens to any addresses
- If the `maxSupply` is not exceeded yet, minters can mint any amount of coins once in every 3 months to any addresses

Recommendation

To improve the trustworthiness of the project, any dynamic runtime updates in the project should be notified to the community. Any plan to invoke the above-mentioned functions should be also considered to move to the execution queue of the Timelock contract.

Alleviation

The development team replied that : We are unable to add the timelock because the shift requirement is real-time. When user wants to shift coins cross chain, e.g. shift from Bsc mainnet to Heco mainnet, they will need to lock like 1000UINC and get tx hash. Then they use the tx hash on BSC and unlock the same amount 1000UINC with tx hash on Heco mainnet. The public can track the shiftment by broswer to https://github.com/UintCola/CrossChainTxPair

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.