Security Assessment

DeTrust

Aug 2nd, 2021

Table of Contents

Summary

This report has been prepared for DeTrust to discover issues and vulnerabilities in the source code of the DeTrust 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	DeTrust
Description	DeTrust is a decentralized trust fund contract.
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/detrust-finance/protocol_v1/tree/main/contracts
Commit	d0744dcada7b3f1406510d69463da7ace48f0142

Audit Summary

Delivery Date	Aug 02, 2021
Audit Methodology	Static Analysis, 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	0	0	0	0	0	0
Informational	3	0	0	3	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IDT	interfaces/IDeTrust.sol	0257da61fdb25fcb8c4c2ed94362fe124061993ad8da0093adef3a01bfe97573
DTE	DeTrust.sol	13882c6c1658c0dbeb186819525cc1ddd357fe02654b7a743ea377de4f8249ba

Findings

Critical	0 (0.00%)
Major	0 (0.00%)
Medium	0 (0.00%)
Minor	0 (0.00%)
Informational	3 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
DeTrust-01	Unlocked Compiler Version Declaration	Language Specific	Informational	Resolved
DeTrust-02	Discussion on Financial Model	Logical Issue	Informational	Resolved
DTE-01	Proper Usage of “public” and “external” type	Coding Style	Informational	Resolved

DeTrust-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	Informational	Global	Resolved

Description

The contract has an unlocked compiler version. An unlocked compiler version in the contract's source code 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 differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify emerging ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

DeTrust team heeded our advice and locked the compiler version as `pragma solidity 0.8.0;`.

This change was applied in commit `fc006e699f48d1be0003ea281adced33c2ca9183`.

DeTrust-02 | Discussion on Financial Model

Category	Severity	Location	Status
Logical Issue	Informational	Global	Resolved

Description

This DeTrust protocol is a decentralized trust fund platform. It will help users host their assets and transfer the assets to a specified beneficiary following their plan. The audited contract currently is a trustworthy protocol. However, it seems that there are no management costs or other administrative fees in this process. What is the plan of the community? We will be glad to discover more information pertaining to this matter from the official website and white paper.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

Alleviation

[DeTrust Team]: We plan to make this protocol free.

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

Category	Severity	Location	Status
Coding Style	Informational	DeTrust.sol: 35	Resolved

Description

`public` functions that are never called by the contract could be declared `external`.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

DeTrust team heeded our advice and declared the function `external`.

This change is applied in commit `7a568608db8dad518bc684af54aa20298c7746d9`.

Appendix

Finding Categories

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.

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 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.