CERTIK

Security Assessment

Decentraland 4

Apr 11th, 2022

Table of Contents

Summary

This report has been prepared for Decentraland to discover issues and vulnerabilities in the source code of the Decentraland 4 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	Decentraland 4
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/decentraland/wearables-contracts/tree/feat/registries
Commit	• 4e6f5198af72445d0fdddbe3dcf9ff7670bb511f • 27d56db8ba2b584be227a6f9d5ac63341abb9065

Audit Summary

Delivery Date	Apr 11, 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	1	0	0	1	0	0	0
Medium	0	0	0	0	0	0	0
Minor	3	0	0	3	0	0	0
Informational	4	0	0	4	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CKP	contracts-addendum-27d56d/oracles	
RWO	contracts-addendum-27d56d/managers/RaritiesWithOracle.sol	7ec5a4910cdab90b5d193b59fb1831611077768e68fba7272c63941fe67229fe
COC	contracts-addendum-27d56d/oracles/ChainlinkOracle.sol	4b2c8038d3f420094298975aee5f3a9c83b76729c64c97053a86fdfdb269b8bb
TPR	contracts/registries/ThirdPartyRegistry.sol	69d8acdddad68b554f3ff6f8ffb94f29297c1b1695393040c7ebc23cc4993b3b
REI	contracts-addendum-27d56d/registries	
TPC	contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol	15d4ca636741dfeabbdb78941406893b37b09f6e3cb931b45a74c6c6b8643204
REG	contracts/registries	

Understandings

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised, which may lead to lost or stolen assets.

There are a few dependent injection contracts and addresses in the current project:

Contract `ThirdPartyRegistry` will set up the following address during initialization:

- `thirdPartyAgregator`: third party aggregator address;
- `feesCollector`: fees collector address;
- `committee`: committee smart contract address;
- `acceptedToken`: accepted ERC20 token address;
- `_owner`: the owner's address.

Contract `RaritiesWithOracle` will set up the following address during initialization:

- `_oracle`: the oracle to acquire "rate";
- `_owner`: the owner's address.

Contract `ChainlinkOracle` will set up the following address during initialization:

- `dataFeed`: the price feeder to acquire "rate".

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privileged Functions

In the contract `ThirdPartyRegistry`, multiple privileged roles are adopted to modify the contract configurations and address attributes:

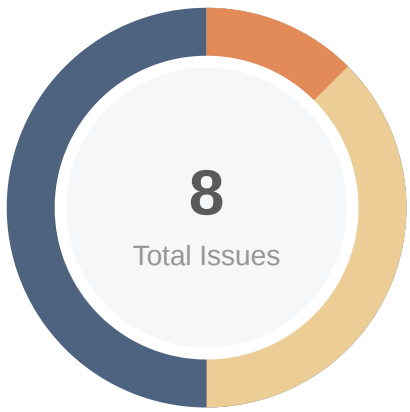
- The role `_owner` is adopted to set different roles and update configurations in the contract.
- The role `thirdPartyAgregator` is adopted to add and update third parties.
- The role `committee` is adopted to review third parties' items and change item approval or content hash status.
- For each third party, the role `manager` is adopted to update third-party settings and add/update items.

In the contract `RaritiesWithOracle`, the role `_owner` is adopted to update the oracles and rarities of the contract.

The advantage of the above roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worthy of note the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

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

Findings



Critical	**0** (0.00%)
Major	**1** (12.50%)
Medium	**0** (0.00%)
Minor	**3** (37.50%)
Informational	**4** (50.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
COC-01	Potential Failure of Rate Calculation	Mathematical Operations	Informational	Acknowledged
CON-01	External Oracle Dependencies	Volatile Code	Minor	Acknowledged
CON-02	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
RWO-01	Lack of Function to Remove Rarities	Logical Issue	Minor	Acknowledged
TPC-01	Lack of Function to Remove ThirdParties and Items	Logical Issue	Minor	Acknowledged
TPC-02	Typos in the Contract	Coding Style	Informational	Acknowledged
TPC-03	Discussion on `initialThirdPartyValue`	Logical Issue	Informational	Acknowledged

CKP-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/registries/ThirdPartyRegistry.sol contracts-addendum-27d56d/managers/RaritiesWithOracle.sol	Acknowledged

Description

The contract `ThirdPartyRegistry` adopted the following privileged roles to enforce the access control during contract runtime:

The role `_owner` has authority over the following functions:

- `setThirdPartyAgregator()` will assign new `thirdPartyAgregator`.
- `setFeesCollector()` will assign new `feesCollector` who will receive fees during `buyItemSlots`(Buy item slots for third parties).
- `setCommittee()` will assign new `committee`.
- `setAcceptedToken()` will setup new accepted ERC20 token during fee collection.
- `setItemTiers()` will set new `itemTiers` contract which estimate the fee for `buyItemSlots`(Buy item slots for third parties).
- `setInitialThirdPartyValue()` will set `initialThirdPartyValue` which will assign a default value that a third party should be init approved or not during `addThirdParties`.
- `setInitialItemValue()` will set `initialItemValue` which will assign a default value that items should be init approved or not during `addItems`.
- `renounceOwnership()` will renounce the ownership of the contract.
- `transferOwnership()` will transfer ownership to a new address.

The role `thirdPartyAgregator` has authority over the following function:

- `addThirdParties()` will add third parties.
- `updateThirdParties()` will update third parties' settings.

The role `committee` member has authority over the following function:

- `reviewThirdParties()` will review third parties' items and change item approval or content hash status.

For each third party, the role `manager` has authority over the following function with corresponding third party:

- `updateThirdParties()` will update third party's settings.
- `addItems()` will add new items to third party.
- `updateItems()` will update new items' metadata.

In the current commit `27d56db8ba2b584be227a6f9d5ac63341abb9065`, the contract `ThirdPartyRegistry` introduce new functions that can be invoked by the privileged roles. The role `owner` has the authority over the following newly introduced functions:

- `setOracle()` will set new oracle contract
- `setItemSlotPrice()` will set the new item slot price

The role `committee` member has authority over the following newly introduced functions:

- `reviewThirdPartyWithRoot()` will review third parties with Merkle Root.
- `setRules()` will set rules for third party.

Additionally, in the contract `RaritiesWithOracle`, the role `_owner` has authority over the following functions:

- `setOracle()` will set new third-party oracle contract.
- `updatePrices()` will update the price for the given rarity by name.
- `addRarities()` will add a list of new rarities.
- `renounceOwnership()` will leave the contract without an owner.
- `transferOwnership()` will transfer ownership to the new owner.

Any compromise to the aforementioned privileged accounts may allow a hacker to take advantage of this authority and manipulate the system.

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 accounts' 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., multi-signature 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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version.

COC-01 | Potential Failure Of Rate Calculation

Category	Severity	Location	Status
Mathematical Operations	Informational	contracts-addendum-27d56d/oracles/ChainlinkOracle.sol: 40	Acknowledged

Description

The function `getRate()` will return the rate in expected decimals,

```
31      function getRate() external view override returns (uint256) {
32          uint256 feedDecimals = uint256(dataFeed.decimals());
33
34          (, int256 rate, , , ) = dataFeed.latestRoundData();
35
36          if (rate <= 0) {
37              revert('ChainlinkOracle#getRate: INVALID_RATE');
38          }
39
40          return uint256(rate).mul(10**(decimals.sub(feedDecimals)));
41      }
```

In L40, it will perform a decimal conversion, the concern is, if `decimals`'s value is smaller than `feedDecimals`, the calculation of `decimals.sub(feedDecimals)` will revert, thus the whole transaction will always fail.

Additionally, there is no function to change `decimals` in the current contract to update the inappropriate `decimals` set during deployment.

Recommendation

In the short term, ensure the interacted oracle's decimals (`feedDecimals`) are always smaller than `decimals`.

In the long term, reconsider the design based on the project logic and ensure if the revert is intended.

Alleviation

[Decentraland]: The team confirmed this is the intended design and no change will be applied in the current version.

CON-01 | External Oracle Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	contracts-addendum-27d56d/oracles/ChainlinkOracle.sol: 1 contracts-addendum-27d56d/managers/RaritiesWithOracle.sol: 1 contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol: 1	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party oracle/data feed protocols to acquire "rate". The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third-party oracles can be compromised and this may lead to lost or stolen assets. In addition, upgrades/updates of third parties can possibly create incompatibility such as decimal changes, protocol changes, etc.

Recommendation

As the project's business logic requires interaction with third-party oracles, it is recommended to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version.

CON-02 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol contracts-addendum-27d56d/oracles/ChainlinkOracle.sol contracts-addendum-27d56d/managers/RaritiesWithOracle.sol	Acknowledged

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 differing compiler version numbers. 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

It is advised that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version over `v0.7.6` the contract should contain the following line:

```
pragma solidity 0.7.6;
```

Alleviation

[Decentraland]: Issue acknowledged. No change will be applied in the current version.

RWO-01 | Lack Of Function To Remove Rarities

Category	Severity	Location	Status
Logical Issue	Minor	contracts-addendum-27d56d/managers/RaritiesWithOracle.sol	Acknowledged

Description

In the contract `RaritiesWithOracle`, the `_owner` is able to add/update "rarity" (array `rarities[]`) via function `addRarities()`/`updatePrices()`. However, there is no equivalent reverse function to remove the "rarities".

The concern is that if a "rarity" is mistakenly updated or a corresponding "rarity" is no longer in use, it cannot be fully removed. Therefore, it could lead to some unexpected errors in the project.

Recommendation

It is advised to add a function to remove the "rarity" that is no longer in use.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version. Additionally, it is the intended design not to remove the rarities.

TPC-01 | Lack Of Function To Remove ThirdParties And Items

Category	Severity	Location	Status
Logical Issue	Minor	contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol	Acknowledged

Description

In the contract `ThirdPartyRegistry`, third parties/items can be added and updated via `addThirdParties()`/`addItems()` and `updateThirdParties()`/`updateItems()`. However, there are no equivalent reverse functions to remove added third parties/items.

The `reviewThirdParties()` will review the value of the third parties/items and update their values. The concern is, that if third parties or items are mistakenly added and reviewed, it will be difficult to remove the third parties/items or mitigate the side effect.

Recommendation

It is advised to add corresponding functions to remove third-parties and items.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version. Additionally, it is the intended design not to remove third parties and items.

TPC-02 | Typos In The Contract

Category	Severity	Location	Status
Coding Style	Informational	contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol	Acknowledged

Description

Variables and comments that contain the `agregator`/`Agregator` field should be replaced with `aggregator`/`Aggregator`.

Recommendation

Recommend revising the typo issue for better readability.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version.

TPC-03 | Discussion On `initialThirdPartyValue`

Category	Severity	Location	Status
Logical Issue	Informational	contracts-addendum-27d56d/registries/ThirdPartyRegistry.sol: 148, 273	Acknowledged

Description

During initialization, the value `initialThirdPartyValue` is set as `true`. However, the `initialItemValue` value is set as `false` in deployment, which is inconsist with the initial value of `initialThirdPartyValue`.

```
147         setInitialItemValue(false);
148         setInitialThirdPartyValue(true);
```

The inconsistency could cause misleading during the review process.

Recommendation

We would like to confirm with the team if this is the intended design.

Alleviation

[Decentraland]: The team acknowledged this issue and decided not to change the codebase in the current version.

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.

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.

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.