CERTIK

Preliminary Comments

decentraland 5

Apr 5th, 2022

Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Understandings

External Dependencies

Privileged Roles

Findings

LAN-01 : Centralization Related Risks

LAN-02 : Users cannot easily revoke approvals on their assets

LAN-03 : Third Party Dependencies

LAN-04 : Missing Access Control for the Function `initialize()`

LAN-05 : Unlocked Compiler Version

LAN-06 : Code Redundancy

LAN-07 : Missing Emit Events

LAN-08 : Potential Reentrancy Issue

LAN-09 : Discussion of the Transfer to `EstateRegistry`

Appendix

Disclaimer

About

Summary

This report has been prepared for decentraland 5 to discover issues and vulnerabilities in the source code of the decentraland 5 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 5
Platform	Ethereum
Language	Solidity
Codebase	https://etherscan.io/address/0x554bb6488ba955377359bed16b84ed0822679cdc#code
Commit	

Audit Summary

Delivery Date	Apr 05, 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	1	0	0	0	0	0
● Medium	1	1	0	0	0	0	0
● Minor	1	1	0	0	0	0	0
● Informational	5	5	0	0	0	0	0
● Discussion	1	1	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
LAN	LANDRegistry.sol	dda075ca939983b95866e8577c620e3e40244ae9c42dea7890bc5f06e840630c

Understandings

Decentraland has created a contract that stores the LAND registry: `LANDRegistry`. The purpose the audit was to audit this contract.

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `IEstateRegistry estateRegistry`, `IMiniMeToken landBalance`, `Storage`, `SafeMath`, `ERC165`.

We assume these vulnerable actors and implementing proper logic to collaborate with the current project.

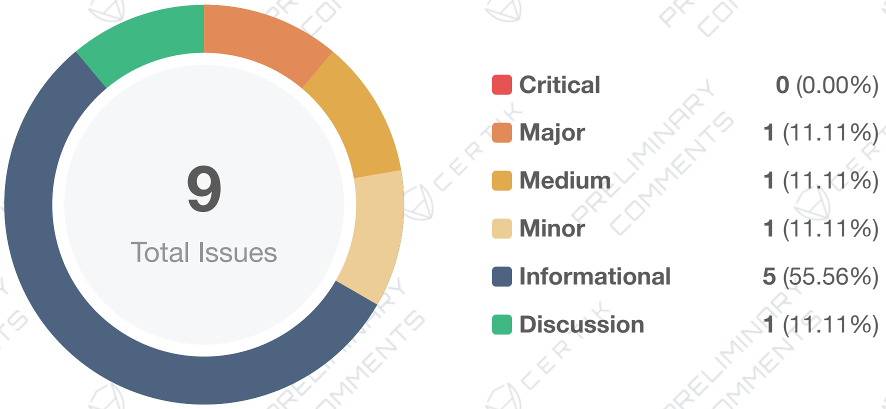
Privileged Roles

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase:

- The `proxyOwner` role is adopted to configure some state variables, and add/remove `deployer` accounts.
- The `deployer` role is adopted to deploy parcels.

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



ID	Title	Category	Severity	Status
LAN-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Pending
LAN-02	Users cannot easily revoke approvals on their assets	Logical Issue	● Medium	ⓘ Pending
LAN-03	Third Party Dependencies	Volatile Code	● Minor	ⓘ Pending
LAN-04	Missing Access Control for the Function `initialize()`	Control Flow	● Informational	ⓘ Pending
LAN-05	Unlocked Compiler Version	Language Specific	● Informational	ⓘ Pending
LAN-06	Code Redundancy	Gas Optimization, Coding Style	● Informational	ⓘ Pending
LAN-07	Missing Emit Events	Language Specific	● Informational	ⓘ Pending
LAN-08	Potential Reentrancy Issue	Logical Issue	● Informational	ⓘ Pending
LAN-09	Discussion of the Transfer to `EstateRegistry`	Logical Issue	● Discussion	ⓘ Pending

LAN-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	LANDRegistry.sol	ⓘ Pending

Description

In the contract `LANDRegistry`, the role `proxyOwner` has authority over the following functions:

- `authorizeDeploy()` : Add a `deployer` address;
- `forbidDeploy()` : Remove a `deployer` address;
- `setEstateRegistry()` : Configure the state variable `estateRegistry`;
- `setLandBalanceToken()` : Modify the state variable `landBalance`.
- `setLatestToNow()`: update the `latestPing` value of a given user.

Any compromise to the `proxyOwner` account may allow a hacker to take advantage of this authority and assign parcels to himself, or disrupt entirely the behavior of the contract.

In the contract `LANDRegistry`, the role `deployer` has authority over the following functions:

- `assignNewParcel()` / `assignMultipleParcels()` : Assign parcel(s) of chosen coordinates to users.

Any compromise to the `deployer` account may allow a hacker to take advantage of this authority and assign parcels to himself.

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., 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.

LAN-02 | Users Cannot Easily Revoke Approvals On Their Assets

Category	Severity	Location	Status
Logical Issue	● Medium	LANDRegistry.sol	ⓘ Pending

Description

Users have the possibility to add `operators` on their assets with the `setApprovalForAll()` function. An operator is a third-party role that is allowed to manage the users assets.

The `operator` has privileged access over the user account; especially, `operators` can approve the assets of the user (with `approve()` function), in order to perform a `_doTransferFrom()`.

In case the `operator` becomes malicious, users can revoke the operator by calling `setApprovalForAll(address operator, false)`.

However, the user will also want to revoke all approvals performed by the malicious operator. Currently, this is not easily possible, users would have to call `approve()` with a new operator on all their assets, which is not intuitive or quick to do.

In case of emergency, users should be able to revoke all approvals on their assets quickly.

Recommendation

A function should be implemented, only callable by the user, in order to easily clear all approvals over their assets.

LAN-03 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	LANDRegistry.sol: 176	ⓘ Pending

Description

The contract is serving as the underlying entity to interact with third party `landBalance`, `estateRegistry` 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.

The functions `landBalance.balanceOf()`, `landBalance.generateTokens()`, `landBalance.destroyTokens()`, `estateRegistry.ownerOf()`, `estateRegistry.mint()` are called in the contract `LANDRegistry`.

Recommendation

We understand that the business logic of `LANDRegistry` requires interaction with `landBalance`, `estateRegistry`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

LAN-04 | Missing Access Control For The Function `initialize()`

Category	Severity	Location	Status
Control Flow	● Informational	LANDRegistry.sol: 964	ⓘ Pending

Description

According to the following codes, the function `initialize()` is used to initialize the value of variables `_name/_symbol/_description` in the contract `LANDRegistry`.

```
function initialize(bytes) external {
  _name = "Decentraland LAND";
  _symbol = "LAND";
  _description = "Contract that stores the Decentraland LAND registry";
}
```

However, in the function `initialize()`, the caller is not checked and the function can be called repeatedly.

As a result, the function `initialize()` can be called by anyone to update the value of these variables after the development team deployed the contract and initialized it.

The impact is however very limited since the variables cannot be modified.

Recommendation

It is recommended to add :

- Access controls over the function `initialize()` ;
- The validation to check if the contract has been initialized.

LAN-05 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	LANDRegistry.sol	ⓘ Pending

Description

The contract has an 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 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.

Additionally, it has been noticed that all contracts are compiled with the compiler version over `0.4.24`, which dates from May 2018. It is recommended to update the compiler versions in the contracts, so they are not exposed to potential security issues related to old compiler versions.

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 `v0.4.24` the contract should contain the following line:

```
pragma solidity 0.4.24;
```

In the long term, if the contracts are compatible with the version `v0.8.0`, it is recommended to use `v0.8.0`.

LAN-06 | Code Redundancy

Category	Severity	Location	Status
Gas Optimization, Coding Style	● Informational	LANDRegistry.sol: 165, 581, 594, 983	ⓘ Pending

Description

The variable `_deprecated_authorizedDeploy` in `LANDStorage` is never used in `LANDRegistry`.

```
165    mapping (address => bool) internal _deprecated_authorizedDeploy;
```

The modifiers `onlyHolder` and `onlyOwnerOf` are defined but never used in the contract:

```
594    modifier onlyHolder(uint256 assetId) {
595      require(_ownerOf(assetId) == msg.sender);
596      _;
597    }
```

```
983    modifier onlyOwnerOf(uint256 assetId) {
984      require(
985        msg.sender == _ownerOf(assetId),
986        "This function can only be called by the owner of the asset"
987      );
988      _;
989    }
```

Additionally, the internal function `_destroy` is not used in the contract:

```
581    function _destroy(uint256 assetId) internal {
582      address holder = _holderOf[assetId];
583      require(holder != 0);
584
585      _removeAssetFrom(holder, assetId);
586
587      emit Transfer(holder, 0, assetId);
588    }
```

Recommendation

It is recommended to remove the redundant codes if it is not intended to be used.

LAN-07 | Missing Emit Events

Category	Severity	Location	Status
Language Specific	● Informational	LANDRegistry.sol: 200	Pending

Description

In the contract `Ownable`, the ownership change does not emit an event to pass the changes out of chain as a notification.

```
198    function transferOwnership(address _newOwner) public onlyOwner {
199      require(_newOwner != owner, "Cannot transfer to yourself");
200      owner = _newOwner;
201    }
```

Recommendation

It is recommended to emit an event in the function `transferOwnership()`, which updates an essential state variable.

LAN-08 | Potential Reentrancy Issue

Category	Severity	Location	Status
Logical Issue	● Informational	LANDRegistry.sol: 690~694	ⓘ Pending

Description

In the `ERC721Base` contract, `_doTransferFrom` will call `_moveToken()` function, which triggers `onERC721Received()` callback of the receiver contract.

```
688      if (doCheck && _isContract(to)) {
689        // Equals to
`bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))
690        require(
691          IERC721Receiver(to).onERC721Received(
692            msg.sender, holder, assetId, userData
693          ) == ERC721_RECEIVED
694        );
695      }
```

However, this external function invocation (`onERC721Received()`) leads to a security loophole. Specifically, the attacker can perform a reentrant call inside the `onERC721Received()` callback.

Note that this will not cause any actual attack in the current audit scope. However, contracts that interact with this contract/function should be aware of the potential reentrancy attack vector.

Recommendation

In the short term, when interacting with this function/contract, follow the check-effect-interaction pattern or use Openzeppelin's "nonReentrant" library.

In the long term, determine if the callback function is required or not. It could be removed to reduce the reentrancy attack vector if it is not intended.

LAN-09 | Discussion Of The Transfer To `EstateRegistry`

Category	Severity	Location	Status
Logical Issue	● Discussion	LANDRegistry.sol: 1183	ⓘ Pending

Description

According to the codes in the function `transferFrom()`, the function is used to transfer the token of the given `assetId`, and the address `to` is checked whether to be estateRegistry.

```
function transferFrom(address from, address to, uint256 assetId) external {
  require(to != address(estateRegistry), "EstateRegistry unsafe transfers are not allowed");
    ...
}
```

Both the functions `transferLand()` and `transferManyLand()` can transfer the token with the `tokenId` encoded the given `x` and `y`. However, the address `to` is not checked here.

The function `transferFrom()` takes the parameter `doCheck` as false when calling `_doTransferFrom()`, differs from the two other functions. In the two others functions, the following codes will trigger if the address `to` is `estateRegistry` and `estateRegistry` is a contract.

```
    if (doCheck && _isContract(to)) {
      // Equals to `bytes4(keccak256("onERC721Received(address,address,uint256,bytes)"))
      require(
        IERC721Receiver(to).onERC721Received(
          msg.sender, holder, assetId, userData
        ) == ERC721_RECEIVED
      );
    }
```

Depending on the implementation of the `estateRegistry`, the transfer would fail when the `IERC721Receiver(to).onERC721Received()` would not succeed. However, it's not sure as `estateRegistry` is out of the scope of the audit and the implementation is unknown here.

Recommendation

The auditors would like to know if this is an intended behavior.

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.

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.

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