CERTIK

Security Assessment

Sandbox - L1

Jul 5th, 2022

Table of Contents

Summary

This report has been prepared for Sandbox - LAND Bridge to discover issues and vulnerabilities in the source code of the Sandbox - L1 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	Sandbox - L1
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/thesandboxgame/sandbox-smart-contracts-private
Commit	• 71ac9c00d2c077e9e8138ec57c6dcd61998129f7 • 2a32f6bab915a969b3e5cec49ba6f3a18c772522

Audit Summary

Delivery Date	Jul 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	0	0	1	0	0	0
Medium	0	0	0	0	0	0	0
Minor	0	0	0	0	0	0	0
Optimization	0	0	0	0	0	0	0
Informational	4	0	0	0	0	0	4
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
LBT	Land/erc721/LandBaseToken.sol	3f970ad1e573cf55bffedf14575e317f6125ebb64a0c31f7d6b8f1b7e5fe8090
ERC	Land/erc721/ERC721BaseToken.sol	36d1644cb034c788de5b5da15ede8f0e9ff0c2e0e485fdf2ef96b50572ae2613
ERM	contracts_common/Interfaces/ERC721MandatoryTokenReceiver.sol	eab3bcbbc4ab4674b288ecdcc55e86f2fe5f9b37a1abd9d95351cb00b6095854
ERT	contracts_common/Interfaces/ERC721TokenReceiver.sol	0d3eae9d617f1afebe8682cc7ff8d04bb87aa99ccc40e979b23ca50f44c176e2
ERE	contracts_common/Interfaces/ERC721Events.sol	cc0c53078e9d9640b95756a7a75678e06766bf50da3d2dc2a94d1580d41db768
ABW	contracts_common/BaseWithStorage/Admin.sol	f336e6bd77e29368a3afe4ffecdc9eafe0b2854f2c303d47405a45a85bfcfb6e
SOB	contracts_common/BaseWithStorage/SuperOperators.sol	307c0411cfc020057e1d38d9ff5a715b088bd074a8351f1cf572fe2b386dfe12
MTR	contracts_common/BaseWithStorage/MetaTransactionReceiver.sol	8bae54108e69e81fcffe22425c311814d7339e078ae37e9c1c67c30cf4e4a6e9
AUL	contracts_common/Libraries/AddressUtils.sol	2a717cd56c8a3f562015bacb0ab7b6d93cb639d64221728520bf3f40217c8957
LKP	Land.sol	3ec4ad7a8a50d8e91986bf6c5f14a9768d5b3d8250e797b8c363994a8eab3e44

Overview

The Sandbox team has implemented the **Land Bridge** functionality. This feature allows the transfer of `LAND` tokens between the root chain (Ethereum) and the child chain (Polygon).

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 and this may lead to lost or stolen assets.

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

- `AddressUtils`, `ERC721Events`, `WithSuperOperators`, and `MetaTransactionReceiver` for the contract `ERC721BaseToken`;
- `ERC721BaseToken` for the contract `LandBaseToken`;
- `LandBaseToken` for the contract `Land`.

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 `LandBaseToken`, the role `_admin` has authority over the following functions:

- `setMinter()`: Decide if an address is a minter or not;

In addition, the `minter` role has authority over the following function:

- `mintQuad()`: mints quads to an address.

Also, the roles `superOperator` and `_metaTransactionContracts` have authority over the following functions:

- `transferQuad()`: Transfer any user's quads to an address;
- `batchTransferQuad()`: Transfer any user's quads to an address.

The contract `LandBaseToken` inherits the contract `ERC721BaseToken`, where `_admin` has authority over the following functions:

- `setMetaTransactionProcessor()`: Give or remove the `_metaTransactionContracts` role to or from an address;
- `setSuperOperator()`: Give or remove the `superOperator` role to or from an address;

- `changeAdmin()`: Change the address of the role `_admin`.

In addition, the `superOperator` role has authority over the following functions:

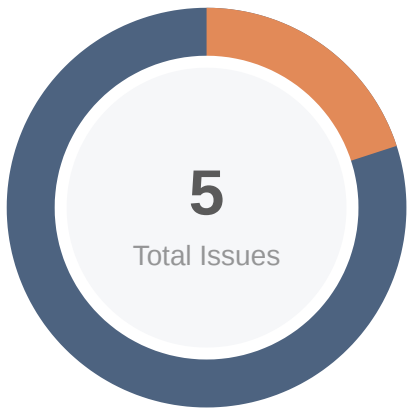
- `approveFor()`: Decide the allowance of any token;
- `approve()`: Decide the allowance of any token;
- `transferFrom()`: Transfer any user's tokens to an address;
- `safeTransferFrom()`: Transfer any user's tokens to an address;
- `batchTransferFrom()`: Transfer several of a user's tokens to an address;
- `safeBatchTransferFrom()`: Transfer several of a user's tokens to an address;
- `setApprovalForAllFor()`: Set the approval for an address to manage all of a user's tokens;
- `burnFrom()`: Burn any user's tokens.

Furthermore, the `_metaTransactionContracts` role has authority over the following functions:

- `approveFor()`: Decide the allowance of any token;
- `transferFrom()`: Transfer any user's tokens to an address;
- `safeTransferFrom()`: Transfer any user's tokens to an address;
- `batchTransferFrom()`: Transfer several of a user's tokens to an address;
- `safeBatchTransferFrom()`: Transfer several of a user's tokens to an address;
- `setApprovalForAllFor()`: Set the approval for an address to manage all of a user's tokens;
- `burnFrom()`: Burn any user's tokens.

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 `Timelock` contract.

Findings



Critical	**0** (0.00%)
Major	**1** (20.00%)
Medium	**0** (0.00%)
Minor	**0** (0.00%)
Informational	**4** (80.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
LBT-01	Inconsistent NatSpec For Minting Function	Inconsistency	Informational	Resolved
LBT-02	Redundant Code	Gas Optimization	Informational	Resolved
LBT-03	Potential Denial-of-Service Attack	Logical Issue	Informational	Resolved
LBT-04	Unable To Mint Burnt Tokens	Logical Issue	Informational	Resolved

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**		Acknowledged

Description

In the contract `LandBaseToken`, the role `_admin` has authority over the following functions:

- `setMinter()`: Decide if an address is a minter or not;

In addition, the `minter` role has authority over the following function:

- `mintQuad()`: mints quads to an address.

Also, the roles `superOperator` and `_metaTransactionContracts` have authority over the following functions:

- `transferQuad()`: Transfer any user's quads to an address;
- `batchTransferQuad()`: Transfer any user's quads to an address.

The contract `LandBaseToken` inherits the contract `ERC721BaseToken`, where `_admin` has authority over the following functions:

- `setMetaTransactionProcessor()`: Give or remove the `_metaTransactionContracts` role to or from an address;
- `setSuperOperator()`: Give or remove the `superOperator` role to or from an address;
- `changeAdmin()`: Change the address of the role `_admin`.

In addition, the `superOperator` role has authority over the following functions:

- `approveFor()`: Decide the allowance of any token;
- `approve()`: Decide the allowance of any token;
- `transferFrom()`: Transfer any user's tokens to an address;
- `safeTransferFrom()`: Transfer any user's tokens to an address;
- `batchTransferFrom()`: Transfer several of a user's tokens to an address;
- `safeBatchTransferFrom()`: Transfer several of a user's tokens to an address;
- `setApprovalForAllFor()`: Set the approval for an address to manage all of a user's tokens;
- `burnFrom()`: Burn any user's tokens.

Furthermore, the `_metaTransactionContracts` role has authority over the following functions:

- `approveFor()`: Decide the allowance of any token;
- `transferFrom()`: Transfer any user's tokens to an address;
- `safeTransferFrom()`: Transfer any user's tokens to an address;
- `batchTransferFrom()`: Transfer several of a user's tokens to an address;
- `safeBatchTransferFrom()`: Transfer several of a user's tokens to an address;
- `setApprovalForAllFor()`: Set the approval for an address to manage all of a user's tokens;
- `burnFrom()`: Burn any user's tokens.

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

Alleviation

[Sandbox]: In the near future, Sandbox will introduce a DAO to decentralize the governance. In the long run, the admin role can also be renounced.

LBT-01 | Inconsistent NatSpec For Minting Function

Category	Severity	Location	Status
Inconsistency	Informational	Land/erc721/LandBaseToken.sol: 68	Resolved

Description

The NatSpec of the `mintQuad()` function states that mints a quad of size 3, 6, 12, or 24 only. However, the function allows mints of size 1.

```
  75      function mintQuad(address to, uint256 size, uint256 x, uint256 y, bytes calldata
data) external {
  76          require(to != address(0), "to is zero address");
  77          require(
  78              isMinter(msg.sender),
  79              "Only a minter can mint"
  80          );
  81          require(x % size == 0 && y % size == 0, "Invalid coordinates");
  82          require(x <= GRID_SIZE - size && y <= GRID_SIZE - size, "Out of bounds");
  83
  84          uint256 quadId;
  85          uint256 id = x + y * GRID_SIZE;
  86
  87          if (size == 1) {
  88              quadId = id;
```

Recommendation

We recommend changing either the NatSpec or the code of `mintQuad()` so that both are consistent with each other.

Alleviation

[Sandbox]: The issue is resolved in commit 2a32f6bab915a969b3e5cec49ba6f3a18c772522 by changing the NatSpec to allow to be consistent with the function.

LBT-02 | Redundant Code

Category	Severity	Location	Status
Gas Optimization	Informational	Land/erc721/LandBaseToken.sol: 327	Resolved

Description

The internal function `_ownerOfQuad()` is used to find the owner or parent owner of a quad and can only be called by itself or one of the regroup functions. As none of these will call `_ownerOfQuad()` with a value of 1 for the input variable `size`, the `if` code branch `size == 1` will never be reached.

Recommendation

We recommend removing the branch `size == 1`.

Alleviation

[Sandbox]: The issue is resolved in commit 2a32f6bab915a969b3e5cec49ba6f3a18c772522 by removing the redundant code.

LBT-03 | Potential Denial-of-Service Attack

Category	Severity	Location	Status
Logical Issue	Informational	Land/erc721/LandBaseToken.sol: 75	Resolved

Description

When the contract tries to mint a quad via `_mintQuad()`, there is a check to ensure that no quads containing the quad to mint and no quads (or LANDS) within the quad to mint have already been minted by calling the function `exists()`.

This leads to a possible denial-of-service attack where the attacker mints 1x1 LANDS at specific locations to prevent the minting of larger quads. For example, out of the possible 166,464 LAND placements, only 289 LANDS need to be minted to prevent 24x24 quads from occurring.

Recommendation

We recommend only allowing mints of larger quads if this is not intended.

Alleviation

[Sandbox]: This is the intended design.

LBT-04 | Unable To Mint Burnt Tokens

Category	Severity	Location	Status
Logical Issue	Informational	Land/erc721/LandBaseToken.sol: 143	Resolved

Description

Minting of tokens on L1 is done by the function `mintQuad()`, which checks whether the tokens already exist or not. In detail, the `mintQuad()` function checks the owner of each token ID to decide whether or not the token has already been minted.

For example, for 1x1 LAND tokens, it checks to see if the owner is non-zero.

```
141         for (uint256 i = 0; i < size*size; i++) {
142             uint256 id = _idInPath(i, size, x, y);
143             require(_owners[id] == 0, "Already minted");
144             emit Transfer(address(0), to, id);
145         }
```

However, for burnt tokens, their value in the `_owners` mapping is non-zero due to the burning flag. This means that burnt tokens cannot be minted again.

Proof of Concept

```
it('Burnt land cannot be minted again', async function () {
  const {
    landContract,
    getNamedAccounts,
    ethers,
    mintQuad,
  } = await setupLand();
  const {deployer, landAdmin} = await getNamedAccounts();
  const contract = landContract.connect(ethers.provider.getSigner(deployer));
  const x = 0;
  const y = 0;
  await mintQuad(deployer, 3, x, y);
  const tokenId = x + y * GRID_SIZE;
  await contract.burn(tokenId);
  await expect(mintQuad(deployer,1,x,y)).to.be.revertedWith('Already minted as 3x3');
});
```

The test passed:

```
    ✓ Burnt land cannot be minted again (64ms)
```

From the result, the burnt token is unable to be minted again.

Recommendation

We recommend allowing burnt tokens to be minted again if this is unintentional.

Alleviation

[Sandbox]: This is the intended design.

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.

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.

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.