CERTIK

Security Assessment

Sandbox - Sand Reward Pool

Feb 4th, 2022

Table of Contents

Summary

This report has been prepared for Sandbox to discover issues and vulnerabilities in the source code of the Sandbox - Sand Reward Pool 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 - Sand Reward Pool
Platform	Polygon
Language	Solidity
Codebase	https://github.com/thesandboxgame/sandbox-smart-contracts/tree/TSBBLOC-445--staking-new-features-on-the-staking-contrat
Commit	• 40a7cf5183560a7ce2f723c58b1d5ab390d01415 • cdd894111b53e37188d0469b0f0ac986229c07a3 • de1b7afe68f7e96866855f9483aba14086325b3c

Audit Summary

Delivery Date	Feb 04, 2022 UTC
Audit Methodology	Static Analysis, Manual Review
Key Components	SandRewardPool

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	0	1	0	0
Medium	1	0	0	0	0	0	1
Minor	1	0	0	0	0	0	1
Informational	3	0	0	0	0	0	3
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SRP	src/solc_0.8/defi/SandRewardPool.sol	4f859edcb281b5494c97b7860b922acaba47ade0acce14ff946b5882d3feaa25
PRC	src/solc_0.8/defi/rewardCalculation/PeriodicRewardCalculator.sol	636f1ba7ff5481df68b399cefbc2333ab9ed35d566233f4827afcf5bea5ab29a
TPR	src/solc_0.8/defi/rewardCalculation/TwoPeriodsRewardCalculator.sol	a327b8c73c50b3c9b57b665b243650c75419392757871d1283876c0268690153
INT	src/solc_0.8/defi/interfaces	
LOC	src/solc_0.8/defi/contributionCalculation/LandOwnerContributionCalculator.sol	5a4ab74b9f172d2bcf354329d405c62e4f607fecbd26a2b30bc38abb3f31b92e
LCC	src/solc_0.8/defi/contributionCalculation/LandContributionCalculator.sol	52042a3227fe416286e3c379567cc87e6a089a8ed0dc1faa5cf9e2a43a4a0f8e
CKP	src/solc_0.8/defi	
IRC	src/solc_0.8/defi/interfaces/IRewardCalculator.sol	2839c3cc3a6da4e36ce29e6954778669178b004b9d2c450609cedb937e628ff8
CCK	src/solc_0.8/defi/contributionCalculation	
STW	src/solc_0.8/defi/StakeTokenWrapper.sol	abdb60ec6d69917ba6f4a23b7d06db0f47b7cb24eee7fbe61c4f20825651a1dc
CCP	src/solc_0.8/defi/rewardCalculation	
ICC	src/solc_0.8/defi/interfaces/IContributionCalculator.sol	b7b57fe1ed5d168314d13f611173caf6f5305f67a40e4e82d7e47fe4b8d84ca5

Overview

The Sandbox team has implemented the Sand staking functionality. According to the Sandbox documentation, two ERC20 compatible tokens are applied in the protocol.

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:

- `multiplierNFToken` for the contract `LandContributionCalculator`;
- `multiplierNFToken` for the contract `LandOwnersAloneContributionCalculator`;
- `rewardPool` for the contract `PeriodicRewardCalculator`;
- `rewardPool` for the contract `TwoPeriodsRewardCalculator`;
- `rewardToken`, `_stakeToken` and `trustedForwarder` for the contract `SandRewardPool`.

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

Privileged Functions

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

- Role `_owner` for the contract `LandContributionCalculator` and `LandOwnersAloneContributionCalculator`;
- Roles `DEFAULT_ADMIN_ROLE`, `REWARD_DISTRIBUTION` and `rewardPool` for the contract `PeriodicRewardCalculator`;
- Roles `DEFAULT_ADMIN_ROLE`, `REWARD_DISTRIBUTION` and `rewardPool` for the contract `TwoPeriodsRewardCalculator`;
- Role `DEFAULT_ADMIN_ROLE` for the contract `SandRewardPool`.

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 a devastating consequence to 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 `Timelock` contract.

Findings

Critical	**0** (0.00%)
Major	**1** (16.67%)
Medium	**1** (16.67%)
Minor	**1** (16.67%)
Informational	**3** (50.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Mitigated
CCP-01	Variable Declare As Immutable	Gas Optimization	Informational	Resolved
SRP-01	Possibility To Bypass The `antiCompoundCheck`	Logical Issue	Medium	Resolved
SRP-02	Missing Input Validation	Volatile Code	Minor	Resolved
SRP-03	Anti-compound System Might Cause An Issue	Logical Issue	Informational	Resolved
STW-01	Incompatibility With Deflationary Tokens	Volatile Code	Informational	Resolved

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**Global**	Mitigated

Description

In the contract `LandContributionCalculator` and `LandOwnersAloneContributionCalculator`, the role `_owner` has authority over the following function:

- `setNFTMultiplierToken()`: Set the `multiplierNFToken` that specifies the amount of users' land.

In the contract `PeriodicRewardCalculator`, the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- `setDuration()`: Modify the duration of the upcoming campaign;
- `grantRole()`/`revokeRole()`: Grant/Revoke a role to/from an account.

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

- `setSavedRewards()`: Modify the variable `savedRewards` and update `lastUpdateTime`;
- `notifyRewardAmount()`: Update the `rewardRate` after a specific amount of reward tokens are sent to the contract.

The `rewardPool` contract has authority over the following functions:

- `restartRewards()`: Reinitiate the values of `lastUpdateTime` and `savedRewards`.

According to the project logic, the `rewardPool` should be initialized as `SandRewardPool` contract, which is a contract. In this case, if correctly initialized, it will not cause any actual issue to the project.

In the contract `TwoPeriodsRewardCalculator`, the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- `grantRole()`/`revokeRole()`: Grant/Revoke a role to/from an account.

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

- `setSavedRewards()`: Set the value of `savedRewards` that are used to calculate rewards;
- `runCampaign()`: Start a staking campaign;
- `setInitialCampaign()`: Start a staking campaign;

- `updateNextCampaign()`: Update parameters of the second period of the current campaign;
- `updateCurrentCampaign()`: Update parameters of the current period of the current campaign;

Additionally, the `rewardPool` contract has authority over the following functions:

- `restartRewards()`: Restart reward (setting `savedRewards = 0`)

According to the project logic, the `rewardPool` should be initialized as `SandRewardPool` contract, which is a contract. In this case, if correctly initialized, it will not cause any actual issue to the project.

In the contract `SandRewardPool`, the role `DEFAULT_ADMIN_ROLE` has authority over the following functions:

- `grantRole()`/`revokeRole()`: Grant/Revoke a role to/from an account.
- `setAntiCompoundLockPeriod()`: Modify the delay between 2 reward withdrawals;
- `setTrustedForwarder()`: Modify the Trusted Forwarder for the Meta Transactions;
- `recoverFunds()`: Send all rewards tokens of contract to an arbitrary destination.
- `setContributionCalculator()`:
- `setRewardToken()`: Modify the reward token;
- `setStakeToken()`: Modify the staked token;
- `setRewardCalculator()`: Modify the contract in charge of calculating rewards.

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]: The team plans to bound all the admin roles to the Sandbox governance multisig. Additionally, multisig will be deployed on polygon in the near future. Since Sandbox is in the process of migrating to Polygon, it is not comfortable enough to use time-lock yet. However, the privileged roles will definitely move to a multi-sig. The Sandbox team is planning to introduce a DAO this year for all the gaming aspects.

CCP-01 | Variable Declare As Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	src/solc_0.8/defi/rewardCalculation/PeriodicRewardCalculator.sol: 33 src/solc_0.8/defi/rewardCalculation/TwoPeriodsRewardCalculator.sol: 62	Resolved

Description

The variable `rewardPool` assigned in the constructor can declare with `Immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since will not be stored in storage. Still, values will be directly inserted the values into the runtime code.

Recommendation

We recommend using an immutable state variable for `rewardPool`.

Alleviation

[Sandbox]: The team heeded the advice and resolved this issue by adding the "immutable" attribute in the commit de1b7afe68f7e96866855f9483aba14086325b3c.

SRP-01 | Possibility To Bypass The `antiCompoundCheck`

Category	Severity	Location	Status
Logical Issue	Medium	src/solc_0.8/defi/SandRewardPool.sol: 71~78	Resolved

Description

The Sandbox has an `anti-compound` mechanism to "implement a time buffer for reward retrieval".

The associated modifier is defined as follow:

```
modifier antiCompoundCheck(address account) {
    require(
        block.timestamp > antiCompound.lastWithdraw[account] +
antiCompound.lockPeriodInSecs,
        "SandRewardPool: must wait"
    );
    antiCompound.lastWithdraw[account] = block.timestamp;
    _;
}
```

The issue is that the variable `antiCompound.lastWithdraw[account]` is not set when a user enters `stake()`, meaning a user does not have to respect the delay when he first enters the staking contract before 7 days.

Below is an example of a scenario that would bypass the mechanism, with `antiCompound.lockPeriodInSecs : 10 080 (7 days)`.

Exploit Scenario:

1. User enters the contract with `stake()`.
   Since `antiCompound.lastWithdraw[account]` is not defined, it is equal to 0.
2. After 2 days, user calls `exit()`, since `antiCompound.lastWithdraw[account]` is 0, the condition below is met, and the user can exit the staking contract.

```
require(block.timestamp > antiCompound.lockPeriodInSecs,"SandRewardPool: must wait");
```

3. The user sends his staking token to a new address, and go back to Step 1.
   The user needs to use a new address since his current address now has a correct value for `antiCompound.lastWithdraw[account]`.

Recommendation

When the `stake()` function is called, if the user is new in the staking contract, the `antiCompound.lastWithdraw[account]` needs to be set to `block.timestamp`.

Alleviation

[Sandbox]: The team heeded the advice and resolved this issue in the commit cdd894111b53e37188d0469b0f0ac986229c07a3.

SRP-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	src/solc_0.8/defi/SandRewardPool.sol: 113~116, 136~139	Resolved

Description

The given input are missing the check for the non-zero address:

- `SandRewardPool.setTrustedForwarder()`: Modify the `Trusted Forwarder`;
- `SandRewardPool.recoverFunds()`: Send rewards tokens from the contract.

Recommendation

It is recommended to add the check for the passed-in values to prevent unexpected error.

Alleviation

[Sandbox]: The team heeded the advice and resolved this issue by applying check in `recoverFunds()` in the commit de1b7afe68f7e96866855f9483aba14086325b3c.

SRP-03 | Anti-compound System Might Cause An Issue

Category	Severity	Location	Status
Logical Issue	Informational	src/solc_0.8/defi/SandRewardPool.sol: 76~77	Resolved

Description

The Sandbox wants to ensure that people stake their tokens for a certain amount of time before being able to withdraw their rewards.

Users can claim their rewards by :

- Calling the `getReward()` function;
- Exiting the staking feature with the `exit()` function.

In both cases, the function to withdraw the rewards is `_withdrawRewards()` :

```
function _withdrawRewards(address account) internal antiCompoundCheck(account) {
```

This `_withdrawRewards()` function uses the `antiCompoundCheck` modifier:

```
modifier antiCompoundCheck(address account) {
    require(
        block.timestamp > antiCompound.lastWithdraw[account] +
antiCompound.lockPeriodInSecs,
        "SandRewardPool: must wait"
    );
    antiCompound.lastWithdraw[account] = block.timestamp;
    _;
}
```

The modifier essentially ensures that a period of time is respected between 2 rewards withdrawals.

The issue is that in case of major security issue regarding the project, users might need to exit the staking immediately with their rewards.
However, if a user does not meet the previous condition from the modifier, the whole `exit()` transaction would revert and the user would be stuck.

The auditors would like to know how The Sandbox team would deal this issue.

Recommendation

In case that this issue appears, the `setAntiCompoundLockPeriod()` could be called with a `lockPeriodInSecs` of zero, which would allow users to withdraw their funds and rewards immediately.

Alleviation

[Sandbox]: The team heeded the advice and resolved this issue in the commit cdd894111b53e37188d0469b0f0ac986229c07a3.

STW-01 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Informational	src/solc_0.8/defi/StakeTokenWrapper.sol: 19~29	Resolved

Description

The staking contract operates as the main entry for interaction with staking users. The staking users deposit tokens into the `SandRewardPool` contract and in return get a proportionate amount of shares. Later on, the staking users can withdraw their own assets from the pool. In this procedure, `_stake()` and `_unstake()` from `StakeTokenWrapper` are involved in transferring users' assets into (or out of) the protocol.

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation

It is recommended to regulate the set of staked tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

As per the protocol documentation, the staking token should be the LP token of the [Sand, ETH] pool, which, if correctly implemented, will not cause actual problem to the project. The status of this issue will be updated after contract deployment upon request.

Alleviation

[SandBox]: The team confirmed that the staking and reward token would be the SAND token which is not a deflationary token.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.