CERTIK

Security Assessment

Coinversation-KACO

Sept 7th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- KTC-01 : Delegation Not Moved Along With Tokens
- KTC-02 : Centralization Risk
- KTC-03 : Function Visibility Optimization
- MCC-01 : Variable Naming Convention
- MCC-02 : Missing Emit Events
- MCC-03 : Unknown Implementation of migrator.migrate()
- MCC-04 : Logic Flaw In `emergencyWithdraw()`
- MCC-05 : Centralization Risk
- MCC-06 : Function Visibility Optimization
- MCC-07 : Incompatibility With Deflationary Tokens
- MCC-08 : `add()` Function Not Restricted

Appendix

Disclaimer

About

Summary

This report has been prepared for Coinversation-KACO to discover issues and vulnerabilities in the source code of the Coinversation-KACO 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	Coinversation-KACO
Description	A token and liquidity pool
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/Coinversation/kaco-farm/blob/283c57506e0a638f64d0a612c4c8b78ec54f7f23/contracts/KacoToken.sol https://github.com/Coinversation/kaco-farm/blob/283c57506e0a638f64d0a612c4c8b78ec54f7f23/contracts/MasterChef.sol https://github.com/Coinversation/kaco-swap-core/blob/56a12afde6e66a3ebf70468fbed443f8d83973ec/contracts/PancakeFactory.sol
Commit	

Audit Summary

Delivery Date	Sept 07, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	5	0	0	5	0	0
Medium	0	0	0	0	0	0
Minor	1	0	0	1	0	0
Informational	5	0	0	5	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
KTC	kaco-farm/KacoToken.sol	805c1fa857d3736b583622ec8de4de10474a0d41ef0ffed2287fe70fe053d574
MCC	kaco-farm/MasterChef.sol	3016ff6b822cacc336f753d9d4ae60e6d346c6a991c64677827fabab866c737e
PFC	kaco-swap-core/PancakeFactory.sol	0ece467faeafbdfd09530b0043bf122a2573e8c3ceb05b789816e722573cf872

It should be noted that the system design includes a number of economic arguments and assumptions. These were explored to the extent that they clarified the intention of the code base, but we did not audit the mechanism design itself. Note that financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol. The correctness of the financial model is not in the scope of the audit.

To bridge the gap of trust between owner and users, the owner needs to express a sincere attitude regarding the considerations of the administrator team's anonymity.

The owner of `KacoToken` has the responsibility to notify users with the following capability in `KacoToken`:

- transfer `Kac` tokens from any account to the contract address account through `lock()`
- mint `Kac` to any account through `mint()`

The owner of `Masterchef` has the responsibility to notify users with the following capability in `PoolStorage`:

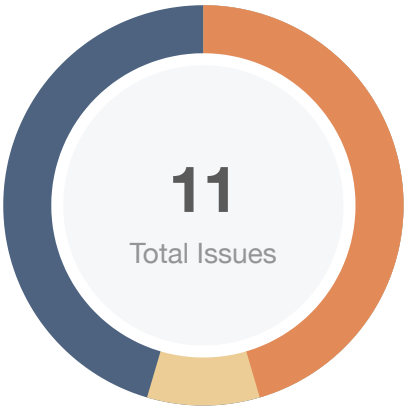
- update key parameters used in reward calculation including `cakePerBlock`, `BONUS_MULTIPLIER`, `allocPoint` of a specific pool through `updateKacPerBlock()`, `updateMultiplier()` and `set()` respectively.
- set the migrator contract through `setMigrator()`

Client response: The `owner` role of `KacoToken` has already been transferred to the `MasterChef` contract so that `lock()` is no longer accessible by anyone and `mint()` can only be called by `MasterChef` for mining rewards. While the `owner` role of the `MasterChef` has been transferred to a timelock contract which would set up a delay period of at least twelve hours for any aforementioned sensitive transactions.

Note that this audit only includes the three contracts listed in the scope exhibit, namely `KacoToken`, `MasterChef` and `PancakeFactory`. The other contracts in the codebase are assumed to be functionally correct and are not audited.

The contract is serving as the underlying entity to interact with third-party PancakeSwap libraries. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness.

Findings



Critical	**0** (0.00%)
Major	**5** (45.45%)
Medium	**0** (0.00%)
Minor	**1** (9.09%)
Informational	**5** (45.45%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
KTC-01	Delegation Not Moved Along With Tokens	Logical Issue	Major	Acknowledged
KTC-02	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
KTC-03	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged
MCC-01	Variable Naming Convention	Coding Style	Informational	Acknowledged
MCC-02	Missing Emit Events	Gas Optimization	Informational	Acknowledged
MCC-03	Unknown Implementation of migrator.migrate()	Logical Issue	Informational	Acknowledged
MCC-04	Logic Flaw In `emergencyWithdraw()`	Logical Issue	Major	Acknowledged
MCC-05	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
MCC-06	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged
MCC-07	Incompatibility With Deflationary Tokens	Volatile Code	Minor	Acknowledged
MCC-08	`add()` Function Not Restricted	Logical Issue	Major	Acknowledged

KTC-01 | Delegation Not Moved Along With Tokens

Category	Severity	Location	Status
Logical Issue	Major	kaco-farm/KacoToken.sol: 48, 83, 95, 114	Acknowledged

Description

The voting power of delegation is not moved from token sender to token recipient along with the `transfer()` and `transferFrom`. Current `transfer()` and `transferFrom()` are from `BEP20` protocol and don't invoke `_moveDelegates()`.

Similarly, the `lock()`, `unlock()` and `transferAll()` functions all move tokens without moving voting power.

Besides, the reimplemented `mint()` function with `_moveDelegates()` should properly override the `mint()` function from `BEP20`.

Recommendation

We advise the client to properly reimplement and override `_transfer()` and `_mint()` functions and always move voting power along with tokens unless there are other considerations.

Alleviation

[Client]: We would not use the current voting modules but will conduct safer community governance by other means.

KTC-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	kaco-farm/KacoToken.sol: 48, 114	Acknowledged

Description

In the contract `KacoToken`, the role `owner` has the authority over the following function:

- `lock()`
- `mint()`

Any compromise to the `owner` account may allow the hacker to take advantage of this and:

- maliciously lock tokens of any account to this contract through `lock()`
- mint `Kaco` tokens to any account through `mint()`

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The client responded that they have already transferred the `owner` role to the `MasterChef` so that `lock()` could not be called by anyone and 'mint()' could only be called by `MasterChef`. Meanwhile `owner` role of `MasterChef` has been transferred to a `Timelock`.

KTC-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	kaco-farm/KacoToken.sol: 28, 36, 40, 44, 83, 95, 120	Acknowledged

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope.

- `unlockedSupply()`
- `totalBalanceOf()`
- `lockOf()`
- `lastUnlockBlock()`
- `unlock()`
- `transferAll()`
- `cap()`

The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

The client responded that since no other contracts call these methods, user's gas consumption would not be affected.

MCC-01 | Variable Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	kaco-farm/MasterChef.sol: 68	Acknowledged

Description

The linked variables do not conform to the standard naming convention of Solidity whereby functions and variable names utilize the format unless variables are declared as constant in which case they utilize the format.

Recommendation

We advise that the naming conventions utilized by the linked statements are adjusted to reflect the correct type of declaration according to the Solidity style guide.

Alleviation

The client responded that they would optimize the naming in the future.

MCC-02 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	kaco-farm/MasterChef.sol: 116	Acknowledged

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

- `setMigrator()`
- `updateMultiplier()`

Recommendation

We advise the client to add events for sensitive actions and emit them in the function.

Alleviation

The client responded that they would call these methods only very rarely, at which they would notify users days in advance. The `owner` role of this contract has been transferred to a time lock so that calling either these methods would incur the `QueueTransactionEvent` emit event and at least twelve-hour delay.

MCC-03 | Unknown Implementation of migrator.migrate()

Category	Severity	Location	Status
Logical Issue	Informational	kaco-farm/MasterChef.sol: 164	Acknowledged

Description

`setMigrator()` function can set migrator contract to any contract that is implemented from `IMigratorChef` interface by the owner. As result, invocation of `migrator.migrate()` in function migrate() may bring dangerous effects as it is unknown to the user. However, the project may lose the ability to upgrade and migrate if `setMigrator()` and `migrate()` are removed. We would like to enquire on the precautions against abusage of the migrate functionality.

Alleviation

The client responded that the migration functions are reserved for special situations and would not be used lightly. The `owner` role of this contract has been transferred to a time lock so that calling either of these methods would incur the `QueueTransactionEvent` event and at least a twelve-hour delay. Besides we would validate our source code before setting `IMigratorChef` and would notify all users in advance in case they want to verify.

MCC-04 | Logic Flaw In `emergencyWithdraw()`

Category	Severity	Location	Status
Logical Issue	Major	kaco-farm/MasterChef.sol: 300	Acknowledged

Description

When `msg.sender` calls `enterStaking()` , `syrup` token will be minted to `msg.sender` when `pool.lpToken` is staked in the contract. However, if the `msg.sender` calls `emergencyWithdraw()`, the `pool.lpToken` can be transferred back to the `msg.sender` but the `syrup` token that has been minted to the `msg.sender` will not be burnt. Therefore, `msg.sender` can call `enterStaking()` and `emergencyWithdraw()` repeatedly to ultimately mint a huge amount of `syrup` tokens, with just the same amount of pool.lpToken.

Recommendation

We advise the client to burn the same amount of `syrup` tokens along with the withdraw of `pool.lpToken` when calling the `emergencyWithdraw()`. i.e:

```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    uint256 userAmount = user.amount;
    user.amount = 0;
    user.rewardDebt = 0;
    syrup.burn(msg.sender, userAmount);
    pool.lpToken.safeTransfer(address(msg.sender), userAmount);
    emit EmergencyWithdraw(msg.sender, _pid, userAmount);
}
```

Alleviation

[Client]: We reviewed this issue and decided that it will not affect our normal functionality because the `syrup` tokens are only used to help demonstrate the user's staking of `KAC` tokens. The `syrup` tokens are worthless by themselves and we would not use them for voting or other things that matter.

MCC-05 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	kaco-farm/MasterChef.sol	Acknowledged

Description

In the contract `MasterChef`, the role `owner` has the authority over the following function:

- `updateKacPerBlock()`
- `updateMultiplier()`
- `add()`
- `set()`
- `setMigrator()`

Any compromise to the `owner` account may allow the hacker to take advantage of this and:

- set `cakePerBlock` without community consent through `updateKacPerBlock()`
- set multiplier without community consent through `updateKacPerBlock()`
- maliciously set allocation points to increase or decrease the reward of a particular pool through `set()`
- add a new lp to the pool without community consent through `add()`
- determine the destination and manner of migration through `setMigrator()`

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Client]: The `owner` role of this contract has been transferred to a time lock so that calling either of these methods would incur the `QueueTransactionEvent` event and at least a twelve-hour delay. Besides, we would soon introduce multi-signature and DAO modules to improve security and transparency.

MCC-06 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	kaco-farm/MasterChef.sol: 108, 116, 126, 141, 153, 158, 216, 238, 259, 280, 300	Acknowledged

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope.

- `updateKacPerBlock()`
- `updateMultiplier()`
- `add()`
- `set()`
- `setMigrator()`
- `migrate()`
- `deposit()`
- `withdraw()`
- `enterStaking()`
- `leaveStaking()`
- `emergencyWithdraw()`

The functions that are never called internally within the contract should have external visibility.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

[Client]: We would improve on visibilities in the future.

MCC-07 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Minor	kaco-farm/MasterChef.sol: 229, 251	Acknowledged

Description

The MasterChef contract operates as the main entry for interaction with staking users. The staking users deposit LP tokens into the pool and in return get a proportionate share of the pool's rewards. Later on, the staking users can withdraw their own assets from the pool. In this procedure, `deposit()` and `withdraw()` 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

We advise the client to regulate the set of LP tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

The client responded that they would review the liquidity providing token to prevent deflationary tokens from being added by one of their pools.

MCC-08 | `add()` Function Not Restricted

Category	Severity	Location	Status
Logical Issue	Major	kaco-farm/MasterChef.sol: 126	Acknowledged

Description

When the same LP token is added into a pool more than once in function `add()`, the total amount of reward in function `updatePool()` will be incorrectly calculated. The current implementation is relying on the operation correctness to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

We advise the client to detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

Alleviation

The client responded that they would review the liquidity providing toke to prevent it from being added repetitively.

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.

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.