CERTIK

Security Assessment

Tokensfarm.com

Dec 1st, 2021

Table of Contents

Summary

This report has been prepared for Tokensfarm.com to discover issues and vulnerabilities in the source code of the Tokensfarm.com 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	Tokensfarm.com
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/Tokensfarm/tokensfarm-contracts/tree/factory/contracts
Commit	4d08b205354abb45852c68c6c0a7ffc23d330795 2637210d25cf6bc31fce940fd649d5cc43b1656b

Audit Summary

Delivery Date	Dec 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	1	0	0	0	0	1
● Major	3	0	0	1	0	2
● Medium	2	0	0	1	0	1
● Minor	7	0	0	1	0	6
● Informational	6	0	0	2	0	4
● Discussion	0	0	0	0	0	0

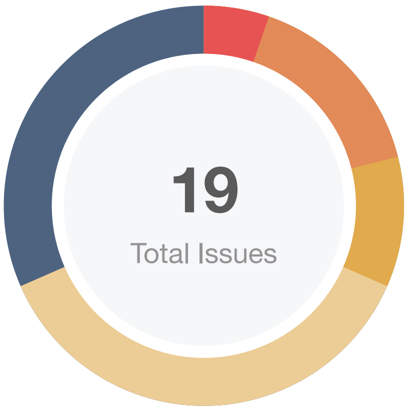
Audit Scope

ID	File	SHA256 Checksum
TFT	TokensFarm.sol	36c2337ba74c3d9e88563c8094baaf2b4f68b463b1b8f668501c9cb65217e3da
TFF	TokensFarmFactory.sol	056779734faa76d42f3cba72fabe2d49c14b0b37aa97b215a804dd8b315a9402

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.

Additionally, 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 accuracy of the financial model is not in the scope of the audit.

Findings



🟥 **Critical**	**1** (5.26%)
🟧 **Major**	**3** (15.79%)
🟨 **Medium**	**2** (10.53%)
🟨 **Minor**	**7** (36.84%)
🟦 **Informational**	**6** (31.58%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Potential Front-Running Risk	Volatile Code	● Minor	ⓘ Acknowledged
GLOBAL-02	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
GLOBAL-03	Missing Emit Events	Gas Optimization	● Informational	✓ Resolved
GLOBAL-04	Lack of Zero Address Validation	Volatile Code	● Minor	✓ Resolved
GLOBAL-05	Address Type Could Be Indexed In Events	Gas Optimization	● Informational	✓ Resolved
TFF-01	Lack of Input Validation	Volatile Code	● Minor	✓ Resolved
TFF-02	Discussion For Contract `TokensFarmFactory`	Logical Issue	● Informational	ⓘ Acknowledged
TFF-03	Discussion For Function `setFeeCollector()`	Logical Issue	● Informational	✓ Resolved
TFT-01	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	✓ Resolved
TFT-02	Multiple Storage Reads	Gas Optimization	● Informational	✓ Resolved
TFT-03	Check Effect Interaction Pattern Violated	Logical Issue	● Minor	✓ Resolved
TFT-04	`totalFeeCollected` Not Cleared	Logical Issue	● Major	✓ Resolved
TFT-05	`totalTokensBurned` Not Updated	Logical Issue	● Medium	✓ Resolved

ID	Title	Category	Severity	Status
TFT-06	Logic Issue Of `totalFeeCollected`	Logical Issue	● Major	✓ Resolved
TFT-07	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	✓ Resolved
TFT-08	Logic Issue Of Function `_erc20Transfer()`	Logical Issue	● Medium	ⓘ Acknowledged
TFT-09	Logic Issue Of Function `withdraw()`	Logical Issue	● Critical	✓ Resolved
TFT-10	Discussion For Function `emergencyWithdraw()`	Logical Issue	● Informational	ⓘ Acknowledged
TFT-11	No Time Limit When Deposit	Volatile Code	● Minor	✓ Resolved

GLOBAL-01 | Potential Front-Running Risk

Category	Severity	Location	Status
Volatile Code	● Minor	Global	ⓘ Acknowledged

Description

Malicious hackers may observe the pending transaction which will execute the `initialize` function, and launch a similar transaction but with the hacker's address of `owner` and gain the ownership of the contract.

For example:

- `TokensFarm.initialize()`
- `TokensFarmFactory.initialize()`

Recommendation

We advise the client to design functionality to only allow a specific user to execute the `initialize` function.

Alleviation

No alleviation.

GLOBAL-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	Global	ⓘ Acknowledged

Description

To bridge the gap in trust between the administrators need to express a sincere attitude regarding the considerations of the administrator team's anonymity.

The `owner` of `TokensFarm` has the responsibility to notify users about the following capabilities:

- set `minTimeToStake` through `setMinTimeToStake()`
- set `isEarlyWithdrawAllowed` through `setIsEarlyWithdrawAllowed()`
- set `stakeFeePercent` through `stakeFeePercent()`
- set `rewardFeePercent` through `setRewardFeePercent()`
- set `flatFeeAmount` through `setFlatFeeAmount()`
- set `isFlatFeeAllowed` through `setIsFlatFeeAllowed()`
- set `feeCollector` through `setFeeCollector()`
- withdraw fee collected in ERC value through `withdrawCollectedFeesERC()`
- withdraw fee collected in ETH value through `withdrawCollectedFeesETH()`
- withdraw stuck tokens on the farm through `withdrawTokensIfStuck()`

The `maintainer` of `TokensFarmFactory` has the responsibility to notify users about the following capabilities:

- deploy and fund farm through `deployAndFundTokensFarm()`
- fund again the farm if necessary through `fundTheSpecificFarm()`
- set `minTimeToStake` in tokens farm through `setMinTimeToStakeOnSpecificFarm()`
- set `isEarlyWithdrawAllowed` in tokens farm through `setIsEarlyWithdrawAllowedOnSpecificFarm()`
- set `stakeFeePercent` in tokens farm through `setStakeFeePercentOnSpecificFarm()`
- set `rewardFeePercent` in tokens farm through `setRewardFeePercentOnSpecificFarm()`
- set `flatFeeAmount` in tokens farm through `setFlatFeeAmountOnSpecificFarm()`
- set `isFlatFeeAllowed` in tokens farm through `setIsFlatFeeAllowedOnSpecificFarm()`
- set `feeCollector` in tokens farm through `setCurrentFeeCollectorOnSpecificFarm()`

The `tokensFarmCongress` of `TokensFarmFactory` has the responsibility to notify users about the following capabilities:

- withdraw fee collected in ERC value through `withdrawCollectedFeesERCOnSpecificFarm()`
- withdraw fee collected in ETH value through `withdrawCollectedFeesETHOnSpecificFarm()`
- withdraw stuck tokens on the farm through `withdrawTokensIfStuckOnSpecificFarm()`
- set `farmImplementation` through `setTokensFarmImplementation()`
- set `feeCollector` through `setFeeCollector()`

Recommendation

We advise the client to carefully manage the privileged account's private keys 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 via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the different levels in terms of the 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

No alleviation.

GLOBAL-03 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	● Informational	Global	✓ Resolved

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers.

For example:

- `TokensFarm.setMinTimeToStake()`
- `TokensFarm.setIsEarlyWithdrawAllowed()`
- `TokensFarm.setStakeFeePercent()`
- `TokensFarm.setRewardFeePercent()`
- `TokensFarm.setFlatFeeAmount()`
- `TokensFarm.setIsFlatFeeAllowed()`
- `TokensFarmFactory.setTokensFarmImplementation()`
- `TokensFarmFactory.setFeeCollector()`

Recommendation

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

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

GLOBAL-04 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	● Minor	Global	✓ Resolved

Description

The given input is missing the check for the non-zero address. For example:

- contract `TokensFarm`: `_beneficiary` in function `withdrawTokensIfStuck()`
- contract `TokensFarmFactory`: `_feeCollector` and `_farmImplementation` in function `initialize()`, `_farmImplementation` in function `setTokensFarmImplementation()`

Recommendation

We advise the client to add the check for the passed-in values to prevent unexpected errors.

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

GLOBAL-05 | Address Type Could Be Indexed In Events

Category	Severity	Location	Status
Gas Optimization	● Informational	Global	✓ Resolved

Description

It is recommended to add `indexed` keyword for parameters in events, which makes it easier for users to navigate event logs.

Recommendation

We advise the client to add keyword `indexed` in the declaration of events.

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFF-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	projects/TokensFarm/contracts/TokensFarmFactory.sol (3772bd0): 485	✓ Resolved

Description

The `start` should less than `end`. If you don't do that there will be underflows.

Recommendation

We advise the client to check that the variables `start` and `end` like as follows:

```
    require(start < end, "start should less than end.");
```

Alleviation

The client heeded our advice and resolved this issue in commit :
0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFF-02 | Discussion For Contract `TokensFarmFactory`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/TokensFarm/contracts/TokensFarmFactory.sol (3772bd0) : 14	ⓘ Acknowledged

Description

Is this only for you or for other partners? If for partners, the `set` operations should only be invoked by the specified owner of the `farm`.

Alleviation

`[TokensFarm]`: It's just for us.

TFF-03 | Discussion For Function `setFeeCollector()`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/TokensFarm/contracts/TokensFarmFactory.sol (3772bd0): 285	✓ Resolved

Description

The function can only change the `feeCollector` of the contract rather than `farm`. We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The client resolved this issue by adding function `setFeeCollector()` and `setCurrentFeeCollectorOnSpecificFarm()` in commit : fbdc555f724255f1689ede4f09e899c39b9471de.

TFT-01 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 553, 612	✓ Resolved

Description

The 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 in the contract. If there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

Alleviation

The client resolved this issue in commit : 88ce173bbeecfd811de38c0c92f5a16cc2f6f8d1.

TFT-02 | Multiple Storage Reads

Category	Severity	Location	Status
Gas Optimization	● Informational	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 310, 333 , 377	✓ Resolved

Description

Repeatedly read from storage, which is very gas inefficient.

Recommendation

We advise the client to assign the values to memory variables first before using, as a call from storage costs 200 gas and a call from memory costs only 3 gas.

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFT-03 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	● Minor	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 612, 690, 722	✓ Resolved

Description

The sequence of external call/transfer and storage manipulation must follow a check effect interaction pattern.

- `withdraw()`
- `emergencyWithdraw()`
- `withdrawCollectedFeesERC()`

Recommendation

We advise the client to adopt the `nonReentrant` modifier from openzeppelin library to the function `emergencyWithdraw()` and `withdraw()` to prevent any reentrancy issue or use the checks-effects-interactions pattern as follows. (LINK)

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFT-04 | `totalFeeCollected` Not Cleared

Category	Severity	Location	Status
Logical Issue	● Major	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 733	✓ Resolved

Description

The function should set `totalFeeCollected` to 0 before calling. If not that, the `owner` can invoke the function more times.

Recommendation

We advise the client to set `totalFeeCollected` to 0.

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFT-05 | `totalTokensBurned` Not Updated

Category	Severity	Location	Status
Logical Issue	● Medium	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 662	✓ Resolved

Description

According to line 656, `totalTokensBurned` should be cumulative when burning to address(1).

Recommendation

We advise the client to update the `totalTokensBurned`.

Alleviation

The client heeded our advice and resolved this issue in commit :

0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFT-06 | Logic Issue Of `totalFeeCollected`

Category	Severity	Location	Status
Logical Issue	● Major	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 75	✓ Resolved

Description

The `totalFeeCollected` records the total fee collected. If the `isFlatFeeAllowed` is true, the `totalFeeCollected` records the amount of ETH, else records the amount of tokens. If the `isFlatFeeAllowed` toggles, the `totalFeeCollected` records sum of ETH and tokens, which results in withdrawing error fee collected in the `withdrawCollectedFeesERC()` or `withdrawCollectedFeesETH()`.

Recommendation

We advise the client to use the different variables to record total fee collected.

Alleviation

The client heeded our advice and resolved this issue in commit : 0623b0a7ee9202fea0ef2da633fc980ba027dd98.

TFT-07 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 488	✓ Resolved

Description

The 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, `fund()` is 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 in the contract. If there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

Alleviation

The client resolved this issue in commit : fbdc555f724255f1689ede4f09e899c39b9471de.

TFT-08 | Logic Issue Of Function `_erc20Transfer()`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 769	ⓘ Acknowledged

Description

When `isFlatFeeAllowed` is `false`, the ether value will be locked in the contract.

Recommendation

We advise the client to recheck the logic.

Alleviation

`[TokensFarm]` : User only pays what we insert on the frontend, so its a non issue..there is no actual scenario where user should send such funds by some hack attempt manually constructing a tx so its ok. we can i. these cases just as dd this to the fees collected eth, but its not really an issue.

TFT-09 | Logic Issue Of Function `withdraw()`

Category	Severity	Location	Status
Logical Issue	● Critical	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 839	✓ Resolved

Description

The function `withdraw()` transfers tokens from the contract to the user. The contract balance after the transfer should be smaller than before. Therefore, The `afterBalance` minus `beforeBalance` causes an underflow error. At the same time, the deduction logic of `totalDeposits` and `stake.amount` is inconsistent.

Recommendation

We advise the client to recheck the logic.

Alleviation

The client resolved this issue in commit : fbdc555f724255f1689ede4f09e899c39b9471de.

TFT-10 | Discussion For Function `emergencyWithdraw()`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 690	ⓘ Acknowledged

Description

There's no fee(msg.value or token) when invoking this. We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

`[TokensFarm]`: No fee is required on the function `emergencyWithdraw()`.

TFT-11 | No Time Limit When Deposit

Category	Severity	Location	Status
Volatile Code	● Minor	projects/TokensFarm/contracts/TokensFarm.sol (3772bd0): 553	✓ Resolved

Description

There is no time when deposit, if someone invokes the deposit function after the `endTime`, it still works.

Recommendation

We advise the client to add a validation for deposit time.

Alleviation

The client resolved this issue in commit : 0de08bc7e4ebcbefdb7394c1410231ba090ef06e.

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.

CERTIK