CERTIK

Security Assessment

Game Coin

Oct 27th, 2021

Table of Contents

Summary

This report has been prepared for Game Coin to discover issues and vulnerabilities in the source code of the Game Coin 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	Game Coin
Platform	BSC
Language	Solidity
Codebase	https://github.com/ericzhu12/Token-Contracts/blob/main/GME.sol https://github.com/TechWallStreet/GMEX-STECH-Development/blob/main/GMEX.sol
Commit	2a6838bde9e12393eadeea73c6d6ac379cb3d26b 84d5717b77d4ad08a6e39d9e9363c79a47cb4549

Audit Summary

Delivery Date	Oct 27, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

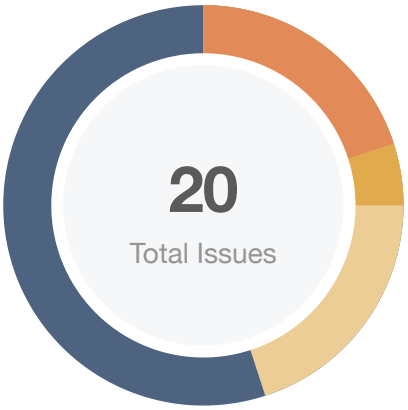
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	4	0	0	2	0	2
Medium	1	0	0	1	0	0
Minor	4	0	0	1	1	2
Informational	11	0	0	2	1	8
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GME	GME.sol	6261ff3dc74e5339ea6e39fb694a9c875d90fb8b0cb2191b065ef9caccd5b600

Findings



Critical	**0** (0.00%)
Major	**4** (20.00%)
Medium	**1** (5.00%)
Minor	**4** (20.00%)
Informational	**11** (55.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GME-01	Unlocked Compiler Version	Language Specific	Informational	Resolved
GME-02	Possible To Gain Ownership After Renouncing The Contract Ownership	**Logical Issue, Centralization / Privilege**	**Major**	Resolved
GME-03	Typo	Coding Style	Informational	Resolved
GME-04	Set `constant` to Variables	Gas Optimization	Informational	Resolved
GME-05	Third Party Dependencies	Control Flow	Minor	Acknowledged
GME-06	The Purpose Of Function `deliver`	Control Flow	Informational	Acknowledged
GME-07	Incorrect Error Message	Logical Issue	Minor	Resolved
GME-08	Return Value Not Handled	Volatile Code	Informational	Resolved
GME-09	Centralized Risk In `addLiquidity`	**Centralization / Privilege**	**Major**	Resolved
GME-10	Function Visibility Optimization	Gas Optimization	Informational	Partially Resolved
GME-11	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
GME-12	Missing Emit Events	Gas Optimization	Informational	Resolved
GME-13	Lack Of Input Validation	Volatile Code	Minor	Partially Resolved
GME-14	Redundant Code	Logical Issue	Informational	Resolved

ID	Title	Category	Severity	Status
GME-15	Visibility Specifiers Missing	Language Specific	Informational	Resolved
GME-16	Potential sandwich attack	Logical Issue	Informational	Acknowledged
GME-17	Potential Reentrancy Risk	Logical Issue	Minor	Resolved
GME-18	Division Before Multiplication	Logical Issue	Informational	Resolved
GME-19	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
GME-20	Not Update `rOwned`	Logical Issue	Medium	Acknowledged

GME-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	GME.sol: 13	Resolved

Description

The contract has an unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-02 | Possible To Gain Ownership After Renouncing The Contract Ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	GME.sol: 405	Resolved

Description

An owner has the possibility to gain ownership of the contract even if he calls function `renounceOwnership` to renounce the ownership. This can be achieved by performing the following operations:

1. Call `lock` to lock the contract. The variable `_previousOwner` is set to the current owner.
2. Call `unlock` to unlock the contract.
3. Call `renounceOwnership` to leave the contract without an owner.
4. Call `unlock` to regain ownership.

Recommendation

We advise the client to update/remove `lock` and `unlock` functions in the contract, or remove the `renounceOwnership` if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound finance as a reference. Reference: https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol

Alleviation

`[CertiK]`: The client has remove the function `renounceOwnership()`

GME-03 | Typo

Category	Severity	Location	Status
Coding Style	Informational	GME.sol: 952, 739, 458	Resolved

Description

There are several typos in the code and comments.

1. In the following code snippet, `tokensIntoLiqudity` should be `tokensIntoLiquidity`.

```
event SwapAndLiquify(
    uint256 tokensSwapped,
    uint256 ethReceived,
    uint256 tokensIntoLiqudity
);
```

2. `recieve` should be `receive` and `swaping` should be `swapping` in the line of comment `//to recieve ETH from uniswapV2Router when swaping`.
3. `geUnlockTime()` should be `getUnlockTime()`.

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-04 | Set `constant` to Variables

Category	Severity	Location	Status
Gas Optimization	Informational	GME.sol: 732, 715, 714, 713, 709	Resolved

Description

The variables `_tTotal`, `numTokensSellToAddToLiquidity`, `_name`, `_symbol` and `_decimals` are unchanged throughout the contract.

Recommendation

We advise the client to set `_tTotal`, `numTokensSellToAddToLiquidity`, `_name`, `_symbol` and `_decimals` as constant variables.

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-05 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	Minor	GME.sol: 723	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party DEX. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised and lead to assets being lost or stolen.

Recommendation

We understand that the business logic of the GME protocol requires interaction DEX for adding liquidity to the GME-BNB pool and swap tokens. We encourage the team to constantly monitor the status of those 3rd parties to mitigate negative outcomes when unexpected activities are observed.

GME-06 | The Purpose Of Function `deliver`

Category	Severity	Location	Status
Control Flow	Informational	GME.sol: 847	Acknowledged

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the GME token balance of the caller by `rAmount`, which is `tAmount` reduces the transaction fee. Then, the function adds `tAmount` to variable `_tFeeTotal`, which represents the contract's total transaction fee. We wish the team could explain more on the purpose of having such functionality.

GME-07 | Incorrect Error Message

Category	Severity	Location	Status
Logical Issue	Minor	GME.sol: 884	Resolved

Description

The error message in `require(_isExcluded[account], "Account is already excluded")` does not describe the error correctly.

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded" .

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-08 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	GME.sol: 1161	Resolved

Description

The return values of function `addLiquidityETH` are not properly handled.

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
);
```

Recommendation

We advise the client to use variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

`[CertiK]`: The client has removed the function.

GME-09 | Centralized Risk In `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	GME.sol: 1166	Resolved

Description

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `GME-BNB` or `GME-ETH` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. 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, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

Alleviation

`[CertiK]`: The client has remove the function `addLiquidityETH()`

GME-10 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	GME.sol: 1033, 947, 929, 925, 873, 856, 847, 843, 839, 834, 829, 823, 818, 814, 809, 800, 796, 792, 788, 471, 463, 458, 452, 443	Partially Resolved

Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope:

- `renounceOwnership()`
- `transferOwnership()`
- `geUnlockTime()`
- `lock()`
- `unlock()`
- `name()`
- `symbol()`
- `decimals()`
- `totalSupply()`
- `transfer()`
- `allowance()`
- `approve()`
- `transferFrom()`
- `increaseAllowance()`
- `decreaseAllowance()`
- `isExcludedFromReward()`
- `totalFees()`
- `deliver()`
- `reflectionFromToken()`
- `excludeFromReward()`
- `excludeFromFee()`
- `includeInFee()`
- `setSwapAndLiquifyEnabled()`
- `isExcludedFromFee()` 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

`[CertiK]`: The function `transfer()` is still declared as `public`.

GME-11 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	GME.sol: 1051, 1047, 947, 941, 937, 933, 929, 925, 903, 896, 883, 873, 463, 452, 443, 1607	Acknowledged

Description

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

- `renounceOwnership()`
- `transferOwnership()`
- `lock()`
- `excludeFromReward()`
- `includeInReward()`
- `addBotToBlacklist()`
- `removeBotFromBlacklist()`
- `excludeFromFee()`
- `includeInFee()`
- `setTaxFeePercent()`
- `setLiquidityFeePercent()`
- `setMaxTxPercent()`
- `setSwapAndLiquifyEnabled()`
- `_setCharityWallet()`
- `_setMarketingWallet()`
- `airdrop()`

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

- renounce ownership through `renounceOwnership()`
- transfer ownership through `transferOwnership()`
- lock the contract through `lock()`
- exclude from reward through `excludeFromReward()`
- include from reward through `includeInReward()`
- add bot to blacklist through `includeInReward()`
- remove bot from blacklist through `includeInReward()`
- exclude from fee through `excludeFromFee()`
- include from fee through `includeInFee()`

- set tax fee percent through `setTaxFeePercent()`
- set liquidity fee percent through `setLiquidityFeePercent()`
- set `_maxTxAmount` through `setMaxTxPercent()`
- enable `swapAndLiquifyEnabled` through `setSwapAndLiquifyEnabled()`
- set `_charityWalletAddress` through `_setCharityWallet()`
- set `_marketingWalletAddress` through `_setMarketingWallet()`
- transfer tokens to anyone through `airdrop()`

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.

GME-12 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	GME.sol: 1051, 1047, 941, 937, 933, 929, 925	Resolved

Description

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

- `excludeFromFee()`
- `includeInFee()`
- `setTaxFeePercent()`
- `setLiquidityFeePercent()`
- `setMaxTxPercent()`
- `_setCharityWallet()`
- `_setMarketingWallet()`

Recommendation

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

```
event TaxFeePercentUpdated(uint256 oldTaxFee, uint256 taxFee);

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    emit TaxFeePercentUpdated(_taxFee,taxFee);
    _taxFee = taxFee;
}
```

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-13 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	GME.sol: 1051, 1047, 748	Partially Resolved

Description

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

Recommendation

We advise the client to add the check for the passed-in values to prevent unexpected error as below:

```
function _setCharityWallet(address payable charityWalletAddress) external onlyOwner() {
    require(charityWalletAddress != address(0), "new charityWallet is the zero address");
    _charityWalletAddress = charityWalletAddress;
}
```

Alleviation

`[CertiK]`: The function `constructor()` is still miss the check for the non-zero address.

GME-14 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	GME.sol: 1180	Resolved

Description

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in `else`.

Recommendation

We advise the client to remove the following code:

```
... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferStandard(sender, recipient, amount);
} ...
```

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-15 | Visibility Specifiers Missing

Category	Severity	Location	Status
Language Specific	Informational	GME.sol: 728	Resolved

Description

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation

`[CertiK]`: The client heeded our advice and resolved this issue.

GME-16 | Potential sandwich attack

Category	Severity	Location	Status
Logical Issue	Informational	GME.sol: 1147	Acknowledged

Description

Potential sandwich attacks could happen if calling `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens` and `uniswapV2Router.addLiquidityETH` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 AToken for 1 ETH, an attacker could raise the price of ETH by adding AToken into the pool before the transaction so we might only get 0.1 ETH. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 ETH.

Recommendation

We advise the client to use Oracle to get an estimation of prices and setting minimum amounts based on the prices when calling the aforementioned functions.

GME-17 | Potential Reentrancy Risk

Category	Severity	Location	Status
Logical Issue	Minor	GME.sol: 1063~1110	Resolved

Description

Function `_transfer()` is risky to reentrancy attack. The function call of `swapAndLiquify()` would eventually send `ethAmount` via `uniswapV2Router.addLiquidityETH()`, and state variable `_rOwned` is massively changed later in the function call of `_tokenTransfer()`. Since the real implementation of the external function is unclear, and the address behind the interface is not clear, reentrancy is possible to take place.

Recommendation

We advise the client to apply OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attacks.

Alleviation

`[CertiK]`: The function `swapAndLiquify()` uses the modifier `lockTheSwap` can do the same thing as `nonReentrant` modifier

GME-18 | Division Before Multiplication

Category	Severity	Location	Status
Logical Issue	Informational	GME.sol: 1132	Resolved

Description

Mathematical operations in the aforementioned function perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We advise the client to apply multiplications before divisions if integer overflow would not happen in functions.

Alleviation

`[CertiK]`: The client has changed the function.

GME-19 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	GME.sol: 751	Acknowledged

Description

`_rTotal` tokens were sent to the `owner` when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team be transparent regarding the initial token distribution process.

GME-20 | Not Update `rOwned`

Category	Severity	Location	Status
Logical Issue	Medium	GME.sol: 883	Acknowledged

Description

When an account is included in the reward list through `includeInReward()`, the account's `rOwned` balance is not updated to reflect the change in `rTotal`. This may lead to a miscalculation of the account's deserved reward. For example, when an account on the reward list is excluded, its `tOwned` balance would be locked for a period of time before the account is included back again. However, in that situation, the universal `rate` is likely to decrease in that period so that by reflecting the `rOwned` the account would receive more tokens than deserved which in effect cancels out that excluded period.

Recommendation

We advise the client to keep `tOwned` unchanged and update `rOwned` accordingly in `includeInReward()`

Alleviation

No alleviation.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.