Security Assessment

Meta Speed

Dec 16th, 2021

Table of Contents

Summary

This report has been prepared for Meta Speed to discover issues and vulnerabilities in the source code of the Meta Speed 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	Meta Speed
Platform	bsc
Language	Solidity
Codebase	https://bscscan.com/address/0x6a0C486139213FDB9c1D4fC1c9f658D0a4e4317E#code
Commit	

Audit Summary

Delivery Date	Dec 16, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

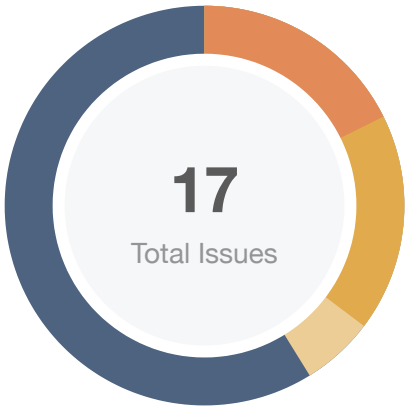
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	3	0	0
Medium	3	0	0	3	0	0
Minor	1	0	0	1	0	0
Informational	10	0	0	10	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
MSM	MetaSpeed/MetaSpeed.sol	cfcef4adab512bae0a0f017fe15e7bc40f91134703384d8758b789c3c42c2c64

Findings



Critical	**0** (0.00%)
Major	**3** (17.65%)
Medium	**3** (17.65%)
Minor	**1** (5.88%)
Informational	**10** (58.82%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
MSM-01	Variable Declare as Immutable	Gas Optimization	Informational	Acknowledged
MSM-02	Variable could be declared as `constant`	Gas Optimization	Informational	Acknowledged
MSM-03	Initial token distribution	**Centralization / Privilege**	**Major**	Acknowledged
MSM-04	Incorrect Error Message	Logical Issue	Minor	Acknowledged
MSM-05	Missing Error Messages	Coding Style	Informational	Acknowledged
MSM-06	Variable `_rOwned[account]` Not Updated in Function `includeInReward()`	Control Flow	Medium	Acknowledged
MSM-07	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
MSM-08	Missing Emit Events	Coding Style	Informational	Acknowledged
MSM-09	Missing Zero Address Validation	Coding Style	Informational	Acknowledged
MSM-10	Lack of Specified Rate Range Restriction	, Logical Issue	Medium	Acknowledged
MSM-11	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
MSM-12	The purpose of function `deliver`	Control Flow	Informational	Acknowledged
MSM-13	Incorrect Burn Token Amount	Logical Issue	Medium	Acknowledged

ID	Title	Category	Severity	Status
MSM-14	Incorrect Transfer Token	Logical Issue	Major	Acknowledged
MSM-15	Discussion For Function `_transferStandard()`	Control Flow	Informational	Acknowledged
MSM-16	Unused Functions	Volatile Code	Informational	Acknowledged
MSM-17	Unused variable	Gas Optimization	Informational	Acknowledged

MSM-01 | Variable Declare as Immutable

Category	Severity	Location	Status
Gas Optimization	Informational	MetaSpeed/MetaSpeed.sol: 161~163, 167, 167	Acknowledged

Description

The variables `_NAME`,`_SYMBOL`, `_DECIMALS` and `_DECIMALFACTOR` 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 these variables.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-02 | Variable could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	MetaSpeed/MetaSpeed.sol: 166	Acknowledged

Description

Variables `_MAX` and `_GRANULARITY` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend declaring those variables as `constant`.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-03 | Initial token distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	MetaSpeed/MetaSpeed.sol: 202	Acknowledged

Description

`_rTotal` of the tokens are sent to the `tokenOwner` when deploying the contract. This could be a centralization risk as the owner can distribute tokens without obtaining the consensus of the community. Since the privilege of the owner, it is possible of being maliciously manipulated by hackers if the account of the owner was compromised.

Recommendation

We recommend the team be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

The team acknowledged this issue and they stated the following:

"The project team will maintain the transparency of token distribution during the project’s marketing and launch process, and all data can be queried on the chain. The private key will not be opened and protect the security of backstage operations. After the project operation is stable, the project team will abandon the Owner's permission, and the community will participate in the common governance."

MSM-04 | Incorrect Error Message

Category	Severity	Location	Status
Logical Issue	Minor	MetaSpeed/MetaSpeed.sol: 310	Acknowledged

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

The team acknowledged this issue and they will leave it as it is for now.

MSM-05 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	MetaSpeed/MetaSpeed.sol: 338	Acknowledged

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We recommend providing the error message.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-06 | Variable `_rOwned[account]` Not Updated in Function `includeInReward()`

Category	Severity	Location	Status
Control Flow	Medium	MetaSpeed/MetaSpeed.sol: 309	Acknowledged

Description

Variable `_rOwned[account]` Not Updated in Function `includeInReward()` The function below has a known bug.

```
309     function includeAccount(address account) external onlyOwner() {
310         require(_isExcluded[account], "Account is already excluded");
311         for (uint256 i = 0; i < _excluded.length; i++) {
312             if (_excluded[i] == account) {
313                 _excluded[i] = _excluded[_excluded.length - 1];
314                 _tOwned[account] = 0;
315                 _isExcluded[account] = false;
316                 _excluded.pop();
317                 break;
318             }
319         }
320     }
```

Variable `_rOwned[account]` is not updated in the function `includeInReward()`, which will make the accounts included siphon off the tokens out of the balances of all token holders.

Details of this finding can be seen in this article from Pera Finance: Link

Recommendation

Variable `_rOwned[account]` Not Updated in Function `includeInReward()` We recommend updating `_rOwned[account]` before setting `_tOwned[account]` to 0.

Sample code:

```
309     function includeInReward(address account) external onlyOwner() {
310         require(_isExcluded[account], "Account is already excluded");
311         for (uint256 i = 0; i < _excluded.length; i++) {
312             if (_excluded[i] == account) {
313                 _excluded[i] = _excluded[_excluded.length - 1];
314                _rOwned[account] = _tOwned[account].mul(_getRate());
315                 _tOwned[account] = 0;
316                 _isExcluded[account] = false;
```

```
317                 _excluded.pop();
318                 break;
319             }
320         }
321     }
```

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-07 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	MetaSpeed/MetaSpeed.sol: 137, 141, 322, 300, 309, 332	Acknowledged

Description

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

- `renounceOwnership()`, to renounce ownership.
- `transferOwnership()`, to transfer ownership.
- `setAsCharityAccount()`, to set charity account.
- `excludeAccount()`, to exclude account.
- `includeAccount()`, to include account.
- `updateFee()`, to update fee.

Any compromise to the `_owner` account may allow the hacker to take advantage of this.

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 team acknowledged this issue and they stated the following:

"The private key will not be opened and protect the security of backstage operations. After the project operation is stable, the project team will abandon the Owner's permission, and the community will participate in the common governance."

MSM-08 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	MetaSpeed/MetaSpeed.sol: 322, 332	Acknowledged

Description

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

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-09 | Missing Zero Address Validation

Category	Severity	Location	Status
Coding Style	Informational	MetaSpeed/MetaSpeed.sol: 186, 322	Acknowledged

Description

Addresses should be checked before assignment to make sure they are not zero addresses.

Recommendation

Consider adding a check as below,

constructor(),

```
require(_FeeAddress != address(0), "fee address cannot be 0");
require(tokenOwner != address(0), "token Owner address cannot be 0");
```

setAsCharityAccount(),

```
require(account != address(0), "charity address cannot be 0");
```

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-10 | Lack of Specified Rate Range Restriction

Category	Severity	Location	Status
logi, Logical Issue	Medium	MetaSpeed/MetaSpeed.sol: 193~195, 332	Acknowledged

Description

The `owner` of the contract has permission to modify the fees without limitation.
Therefore, in the extreme case, that fee could be a very large amount of value, which might cause unexpected loss to the project and users.

Recommendation

We advise the client to set a reasonable range restriction for the aforementioned states to ensure the fair distribution of the fees/tokens.

Alleviation

The team acknowledged this issue and they stated the following:

"They will set a fixed value before the start of marketing, and every function change will take a community vote and other plans to ensure the fair distribution of fees."

MSM-11 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	MetaSpeed/MetaSpeed.sol: 14	Acknowledged

Description

The contract has 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 an 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

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.2` the contract should contain the following line:

```
pragma solidity 0.8.2;
```

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-12 | The purpose of function `deliver`

Category	Severity	Location	Status
Control Flow	Informational	MetaSpeed/MetaSpeed.sol: 274	Acknowledged

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the MetaSpeed 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.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-13 | Incorrect Burn Token Amount

Category	Severity	Location	Status
Logical Issue	Medium	MetaSpeed/MetaSpeed.sol: 337	Acknowledged

Description

The `burn()` function is used to burn the reflection tokens by the amount specified by `_value` parameter. The amount of reflection tokens burnt needs to be converted(amount = amount * _getRate()) to the decrease in normal token before it is then subtracted from the total token amount `_rOwned`. Besides, if the user-specified by `_who` argument has been excluded from the reward, the account's `tOwned` should decrease by `_value`. The current bookkeeping logic is inconsistent with the underlying asset shift.

Recommendation

We advise the client to review the function again.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-14 | Incorrect Transfer Token

Category	Severity	Location	Status
Logical Issue	Major	MetaSpeed/MetaSpeed.sol: 362	Acknowledged

Description

When the desired address is excluded from receiving holder rewards, it cannot benefit from the deflation, `_tOwned` tracks the token amount owned by the address. The sender transfers tokens to the recipient which is excluded from the rewards, it will get incorrect results when calling the function `balanceOf()`.

Recommendation

We advice the client to review the function again.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-15 | Discussion For Function `_transferStandard()`

Category	Severity	Location	Status
Control Flow	Informational	MetaSpeed/MetaSpeed.sol: 366	Acknowledged

Description

The function `_transferStandard()` is useless to call `_reflectFees()`, so the transaction fee will not be deducted during the transfer process, and there will be no deflation. We would like to know if it is consistent with project design?

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-16 | Unused Functions

Category	Severity	Location	Status
Volatile Code	Informational	MetaSpeed/MetaSpeed.sol: 378, 390, 402	Acknowledged

Description

The private functions `_transferToExcluded()`、`_transferFromExcludeds()` and `_transferBothExcluded()` are never used.

Recommendation

We advise removing it if there is no plan for further usage.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

MSM-17 | Unused variable

Category	Severity	Location	Status
Gas Optimization	Informational	MetaSpeed/MetaSpeed.sol: 149	Acknowledged

Description

The variable `isMint` on line 15 is declared but never used.

Recommendation

We recommend removing the unused variable.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

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.