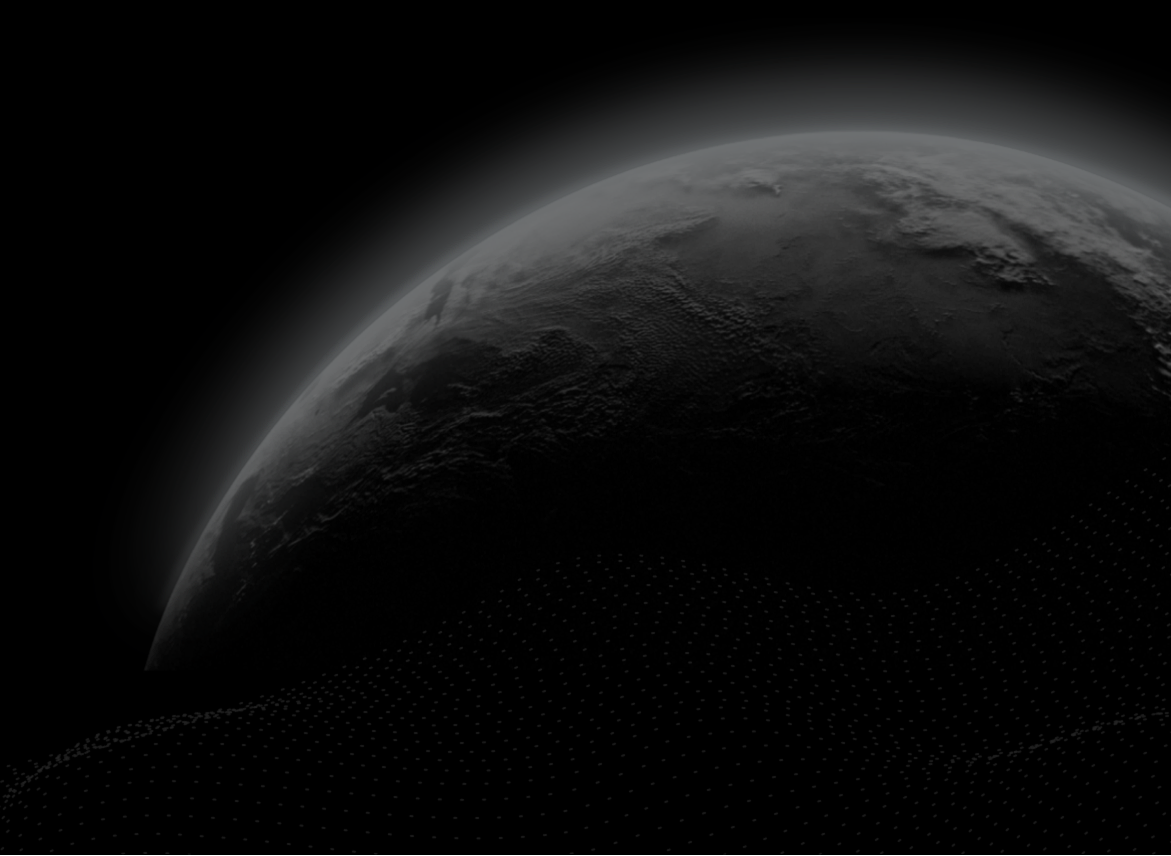
Economic Assessment

T.I.M.E. Dividend(TIME) - Polygon

CertiK Assessed on Dec 8th, 2023



CertiK Assessed on Dec 8th, 2023

T.I.M.E. Dividend(TIME) - Polygon

The economic assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Polygon (MATIC)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 12/08/2023	N/A

CODEBASE

https://polygonscan.com/token/0x9F42bcA1A579fCf9Efc165a0244B129
37e18C6A5

View All in Codebase Page

Vulnerability Summary

Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined
0	0	0	0	0	0

Count	Severity	Description
0	Critical	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
0	Major	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
0	Medium	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
0	Minor	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
0	Informational	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | T.I.M.E. DIVIDEND(TIME) - POLYGON

CODEBASE | T.I.M.E. DIVIDEND(TIME) - POLYGON

Repository

https://polygonscan.com/token/0x9F42bcA1A579fCf9Efc165a0244B12937e18C6A5

AUDIT SCOPE | T.I.M.E. DIVIDEND(TIME) - POLYGON

1 file audited ● 1 file without findings

ID	Repo	File	SHA256 Checksum
● TIM	mainnet	contracts/TIMEDividend.sol	610663a652d489d47d40e682cd0e794827ea6a4617b0297c9dc688bc85090d2d

APPROACH & METHODS | T.I.M.E. DIVIDEND(TIME) - POLYGON

This report has been prepared for T.I.M.E. Dividend to discover issues and vulnerabilities in the source code of the T.I.M.E. Dividend(TIME) - Polygon project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis 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:

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

INTRODUCTION | T.I.M.E. DIVIDEND(TIME) - POLYGON

The `TIMEDividend` contract allows for the distribution of dividends to token holders. The dividends are paid out in native coins (MATIC), with the amount distributed based on the number of tokens held by each address after delta correction. The delta correction moves opposite to the token flow of token transferring, such that in general, the dividend distribution is aligned with the initial token holding status. Generally we do not recommend the token distribution to have more than half of the total supply held by one user, given that the initial token distribution takes place before the contract is ready and allowed to work.

The contract uses a unique approach to calculate dividends, where `magnifiedDividendPerShare` and `magnifiedDividendCorrections` of each address are to ensure that the dividend payouts/claims are maintained over time.

A core value of the calculation if the state variable `magnitude`, which is a constant value used to convert amounts to scaling magnitudes. It used to maintaining the resolution of payouts to be accurately calculated for very small amounts. It is hardcoded to 2^{128} in the contract.

The contract contains two key mappings, `cumulativeDividendClaimed` and `magnifiedDividendCorrections`. `cumulativeDividendClaimed` is used to track the cumulative amount of dividend claimed by each address, ensuring that double payouts are not made. `magnifiedDividendCorrections` is used to track corrections made to the magnified dividend per share as tokens are transferred between accounts.

The `receive()` function is aim to receive fees generated from the swap operations, which is not implemented in the TIMEDividend contract. In fact the `receive` function does not specify which address is the source of the fees, such that it allows any addresses to send native coin (MATIC) to itself. The function requires that the minting process is complete and the ownership has been renounced, which can also be seen as a status that the whole contract is ready to start functioning.

PROTOCOL DESCRIPTION | T.I.M.E. DIVIDEND(TIME) - POLYGON

State Variables

```
uint256 public constant magnitude = 2**128;
uint256 public magnifiedDividendPerShare;

mapping(address => int256) public magnifiedDividendCorrections;
mapping(address => uint256) public cumulativeDividendClaimed;
```

Functions

receive()

Let a_b be the native coins (MATIC) transfer amount, which is also known as `msg.value` in Solidity. For each function call, we have

$$magnifiedDividendPerShare += \frac{a_b}{totalSupply} \times magnitude$$

If the receive function is called for n times, we have

$$magnifiedDividendPerShare_n = \sum_{i=1}^{n} \frac{a_{bi}}{totalSupply} \times magnitude$$

$$\implies magnifiedDividendPerShare_n = 2^{128} * \sum_{i=1}^{n} \frac{a_{bi}}{totalSupply}$$

where `totalSupply` cannot be increased since the require statement of the `receive` function checks that the contract ownership is already renounced.

_beforeTokenTransfer()

Let's say there is a transfer transaction, where u_s is the sender's address, u_r is the recepient address, and a_t is the token transfer amount. Let `magnifiedDividendCorrections` be mdc. If this function is called for n times, we have

$$mdc[u_s] = \sum_{i=1}^{n} magnifiedDividendPerShare \times a_t$$

and

$$mdc[u_r] = -\sum_{i=1}^{n} magnifiedDividendPerShare \times a_t$$

divideFrom()

$$product = magDividendPerShare * balance + correction$$

$$return_1 = product/magnitude$$

$$\implies return_1 = (magDividendPerShare * balance + correction) \div magnitude$$

$$return_2 = product \mod magnitude$$

$$\implies return_2 = (magDividendPerShare * balance + correction) \mod magnitude$$

accumulativeDividendOf()

Let `magnifiedDividendCorrections` be mdc, and `account` be the input address. Also since there are two parts of the return value, let the former value be $return_1$ and the latter value be $return_2$.

$$return_1 = product/magnitude$$

$$\implies return_1 = (magDividendPerShare * balanceOf(account) + mdc[account]) \div magnitude$$

$$return_2 = product \mod magnitude$$

$$\implies return_2 = (magDividendPerShare * balanceOf(account) + mdc[account]) \mod magnitude$$

claimableDividendOf()

Let `magnifiedDividendCorrections` be mdc, `account` be the input address, and `cumulativeDividendClaimed` be cdc, we have

$$return = (return_1 \ of \ dividendFrom) - cdc[account]$$

$$\implies return = \frac{magDividendPerShare*balanceOf(account)+mdc[account]}{magnitude} - cdc[account]$$

claimDividend()

Let `magnifiedDividendCorrections` be mdc, and let `cumulativeDividendClaimed` be cdc.

$$claimable = \frac{magDividendPerShare*balanceOf(account)+mdc[account]}{magnitude} - cdc[account]_{old}$$

$$recipent \ balance += claimable \text{ , where currency MATIC}$$

$$cdc[account] += claimable$$

distributeAll()

This function is removed in commit hash `d6c89e5dac14b6db95f9dc67af54bd76103805fe`.

~~Called function `distributeAll()` from interface `IInternetMoneySwapRouter`. The function sends all fees, the input `amount` of native coins and/or WETH tokens, to the `destination` address defined in the contract behind the `IInternetMoneySwapRouter`.~~

PROTOCOL ANALYSIS | T.I.M.E. DIVIDEND(TIME) - POLYGON

The smart contract and its functions don't maintain any time-related variables, so the length of time a user holds TIME tokens doesn't affect the final dividend amount. We thoroughly examined the state and local variables of the TIMEDividend contract and found that it doesn't store or use any external data related to a locking time period. Therefore, the only variables that influence a user's dividend/reward are the `magnifiedDividendPerShare` , the amount of TIME tokens held by the user's address, and the `magnifiedDividendCorrections` (mdc) of the user's address.

Claimable Dividend

Within the four state variables, `magnitude` is declared to be `constant` .

`magnifiedDividendPerShare` is a variable that keeps track of the magnified dividend per share. It is calculated by dividing the total amount of dividend received by the `<total supply of tokens>` , and then multiplying by `magnitude` (2^{128}).

The `magnifiedDividendCorrections` mapping keeps track of the magnified dividend corrections for each account. Magnified dividend corrections are used to adjust the claimable dividend of an account based on its transfer history.

The `cumulativeDividendClaimed` mapping keeps track of the cumulative dividend claimed for each account. It is used to calculate the total claimable dividend for an account.

Here we would like to summarize a general math expression of the claimable dividend of a user. For the n-th time the function `claimDividend` is being called by an address, define the follow variables:

- `msg.sender` , the function caller address: u
- `magnifiedDividendPerShare` : `mdps`
- the previous claimed dividend summation: cdc_{n-1}
- the number of function calls of `claimDividend` before this call : n_c
- the number of function calls of `receive` : n_{nc}
  - the received MATIC amount of the i_{nc} time with the total n_{nc} time:$amount_{i_{nc}}$
- the number of function calls of `transfer` as a sender: n_{ts}
  - the sent token amount of the i_{ts} time with the total n_{ts} time: $amount_{i_{ts}}$
- the number of function calls of `transfer` as a receiver: n_{tr}
  - the received token amount of the i_{tr} time with the total n_{tr} time: $amount_{i_{tr}}$
- the number of function calls of `burn` : n_b
  - the burnt token amount of the i_b time with the total n_b time: $amount_{i_b}$
- initial token balance of the user: $initBal$

- current token balance of the user: $currBal$, at the n-th call of `claimDividend`

From the above function description, we have

$$claimable_n = \frac{magDividendPerShare_{n_{nc}} * balanceOf(account) + mdc[account]}{magnitude} - cdc_{old}$$

Here for the balance of u at the n-th call of `claimDividend`, the current token balance is

$$currBal = initBal - <\text{all sent amount}> + <\text{all received amount}> - <\text{all burnt amount}>$$

$$\implies currBal = initBal - \sum_{i_{ts}=1}^{n_{ts}} amount_{i_{tr}} + \sum_{i_{ts}=1}^{n_{tr}} amount_{i_{tr}} - \sum_{i_b=1}^{n_b} amount_{i_b}$$

Similarly, we have the `magnifiedDividendCorrections` be

$$mdc = mdps * (<\text{all sent amount}> - <\text{all received amount}> + <\text{all burnt amount}>)$$

$$\implies mdc = mdps * (\sum_{i_{ts}=1}^{n_{ts}} amount_{i_{tr}} - \sum_{i_{ts}=1}^{n_{tr}} amount_{i_{tr}} + \sum_{i_b=1}^{n_b} amount_{i_b})$$

In the meanwhile, `magnifiedDividendPerShare` is monotonically increasing controlled by the `receive` function. From the above function description of `receive`, we have

$$magnifiedDividendPerShare_{n_{nc}} = \sum_{i_{nc}=1}^{n_{nc}} \frac{amount_{i_{nc}}}{totalSupply} \times magnitude$$

Also, for the previous claimed dividend summation, we have $cdc_{n-1} = \sum_{i=1}^{n-1} claimable_i$

Therefore, for $claimable_n$, we have

$$claimable_n = \frac{magDividendPerShare_{n_{nc}} * balanceOf(account) + mdc[account]}{magnitude} - cdc_{old}$$

Substitute the variable names and make them fit the latest definition in the analysis.

$$claimable_n = \frac{mdps_{n_{nc}} * currBal + mdc}{magnitude} - cdc_{n-1}$$

Since $mdc == mdps * (<\text{transfer amount delta}>)$, we can extract $\frac{mdps}{magnitude}$, and then we have

$$claimable_n = \frac{mdps}{magnitude} * (currBal + \frac{mdc}{mdps}) - \sum_{i=1}^{n-1} claimable_i$$

Substitute $currBal$ and mdc, we have the expression with the detailed amount summation based on the times of different functions being called for the current `receiver` function caller.

$$\implies claimable_n = \frac{mdps}{magnitude} * (initBal - \sum_{i_{ts}=1}^{n_{ts}} amount_{i_{tr}} + \sum_{i_{ts}=1}^{n_{tr}} amount_{i_{tr}} - \sum_{i_b=1}^{n_b} amount_{i_b} + \sum_{i_{ts}=1}^{n_{ts}} amount_{i_{tr}} - \sum_{i_{ts}=1}^{n_{tr}} amount_{i_{tr}} + \sum_{i_b=1}^{n_b} amount_{i_b}) - \sum_{i=1}^{n-1} claimable_i$$

$$= \frac{mdps}{magnitude} * initBal - \sum_{i=1}^{n-1} claimable_i$$

$$= \frac{\sum_{i_{nc}=1}^{n_{nc}} \frac{amount_{i_{nc}}}{totalSupply} \times magnitude}{magnitude} - \sum_{i=1}^{n-1} claimable_i$$

$$= \sum_{i_{nc}=1}^{n_{nc}} \frac{amount_{i_{nc}}}{totalSupply} - \sum_{i=1}^{n-1} claimable_i$$

Here when $i = 1$, the base case gives that the $claimable_1 = 0$, and the first time claimable dividend is the sum of quotient of each native coin (MATIC) deposit divided by the total supply at that time.

APPENDIX | T.I.M.E. DIVIDEND(TIME) - POLYGON

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 | CERTIK

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.

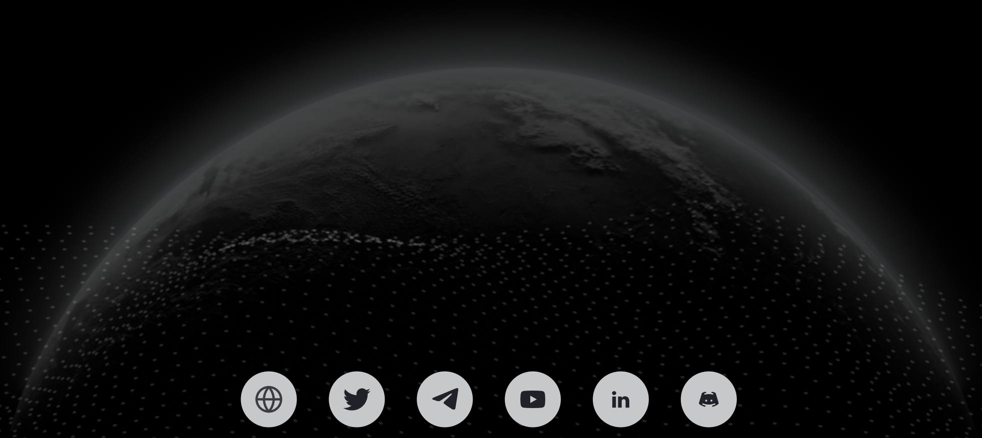
NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND 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.

CertiK | Securing the Web3 World

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.



 Copyright © CertiK