CERTIK

Security Assessment

ShivaToken

Oct 28th, 2021

Table of Contents

Summary

This report has been prepared for ShivaToken to discover issues and vulnerabilities in the source code of the ShivaToken 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	ShivaToken
Description	BEP20 and Dividend Token
Platform	BSC
Language	Solidity
Codebase	https://github.com/ShivaToken/ShivaToken
Commit	a36effc2c20f1526e02a642e565a0caa2b946322

Audit Summary

Delivery Date	Oct 28, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

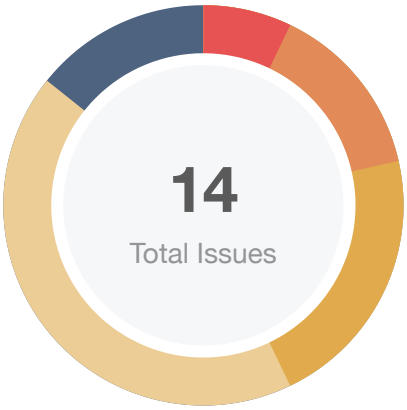
Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
STS	ShivaToken.sol	d618a633197c9be94523be215a947cf7c04dd01a8aafe52c1f28efc959ac98d2

Findings



Critical	1 (7.14%)
Major	2 (14.29%)
Medium	3 (21.43%)
Minor	6 (42.86%)
Informational	2 (14.29%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
STS-01	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved
STS-02	Fees Stored in Contract Withdrawable By Owner	**Centralization / Privilege**	**Critical**	Partially Resolved
STS-03	No Upper Limits for Fees	**Centralization / Privilege**	**Major**	Resolved
STS-04	Potential Reentrancy Attack	Logical Issue	Medium	Resolved
STS-05	`deadWallet` Not Excluded From Dividends	Logical Issue, Inconsistency	Minor	Resolved
STS-06	`_uniswapV2Pair` Not Set As Automated Market Maker Pair	Logical Issue, Inconsistency	Medium	Resolved
STS-07	`_marketingWalletAddress` State Not Consistent on Change	Logical Issue, Inconsistency	Minor	Resolved
STS-08	Usage of `transfer()` for sending BNB	Volatile Code	Minor	Resolved
STS-09	Gas Fee Passed to User	Volatile Code, Control Flow	Minor	Acknowledged
STS-10	Return Value Ignored	Volatile Code	Minor	Resolved
STS-11	Requirement Always Passes	Gas Optimization	Medium	Resolved
STS-12	Variable Declaration as `constant`	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
STS-13	Missing Emit Events	Coding Style	Informational	Resolved
STS-14	Inconsistency With White Paper	Inconsistency	Minor	Acknowledged

STS-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	ShivaToken.sol: 377, 386, 1101, 1375, 1392, 1401, 1408, 1415, 1420, 1425, 1431, 1437, 1443, 1448, 1453, 1459, 1463, 1467, 1471, 1476, 1481, 1486, 1492, 1507, 1514, 1518, 1554, 1842, 1851, 1368, 1611, 1615, 1649, 1650	Partially Resolved

Description

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

- `renounceOwnership()` which revokes ownership
- `transferOwnership()` which sets `_owner`
- `distributeBTCBDividends()` which increases `magnifiedDividendPerShare` and `totalDividendsDistributed`
- `updateDividendTracker()` which sets `dividendTracker`
- `updateUniswapV2Router()` which sets `uniswapV2Pair`
- `excludeFromFees()` which sets `_isExcludedFromFees[account]`
- `excludeMultipleAccountsFromFees()` which calls `excludeFromFees()` for multiple addresses
- `setExcludedFromAntiWhale()` which sets `_excludedFromAntiWhale[accounts]`
- `setExcludedFromLimitSwap()` which sets `_excludedLimitSwap[accounts]`
- `updateMaxTransferAmountRate()` which sets `maxTransferAmountRate`
- `updateMaxBuyAmount()` which sets `maxBuyAmount`
- `updateMaxSellAmount()` which sets `maxSellAmount`
- `UpdateLimitSwap()` which sets `limitSwap`
- `updateSwapAndLiquifyDividendEnabled()` which sets `swapAndLiquifyDividendEnabled`
- `UpdateTimeLimitSwap()` which sets `timeLimitSwap`
- `setSelling()` which sets `selling`
- `setBuying()` which sets `buying`
- `setMarketingWallet()` which sets `_marketingWalletAddress`
- `setBTCBRewardsFee()` which sets `BTCBRewardsFee`
- `setLiquiditFee()` which sets `liquidityFee`
- `setMarketingFee()` which sets `marketingFee`
- `setAutomatedMarketMakerPair()`
- `blacklistAddress()` which sets `_isBlacklisted[account]`
- `updateGasForProcessing()` which sets `gasForProcessing`

- `withdrawShiva()` which `transfers()` `amount` tokens to `toAddress`
- `withdrawBNB()` which `transfers()` `amount` BNB to `toAddress`
- `excludeFromDividends()` which removes `account` from dividends, clearing balance
- `updateClaimWait()` which sets `claimWait`
- `updateMinimumTokenBalanceForDividends()` which sets `minimumTokenBalanceForDividends`

As well as:

- reception of 51,000,000,000 tokens minted to on construction
- exemption from buying and selling `_transfer()` restrictions
- exemption from triggering any swaps or fees

Any compromise to the `_owner` account may allow the hacker to take advantage of this and drastically affect the contract state.

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

[ShivaToken Team]: updated source code includes yam governance, however the `_owner` account will still need to be monitored until ownership is transferred.

STS-02 | Fees Stored in Contract Withdrawable By Owner

Category	Severity	Location	Status
Centralization / Privilege	**Critical**	ShivaToken.sol: 1687, 1842	Partially Resolved

Description

The fees generated in the function `_transfer()` are stored in the contract until the conditions on lines 1645 to 1650 are satisfied. The `_owner` is able to transfer the Shiva Token accumulated from fees using function `withdrawShiva()` to an arbitrary address.

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.

As well as potentially distribute fees to intended recipients immediately during token transfers.

Alleviation

[ShivaToken Team]: updated source code includes yam governance, however the `_owner` account will still need to be monitored until ownership is transfered.

STS-03 | No Upper Limits for Fees

Category	Severity	Location	Status
Centralization / Privilege	**Major**	ShivaToken.sol: 1472, 1477, 1482	Resolved

Description

There are no upper limits restricting `BTCBRewardsFee`, `liquidityFee`, `marketingFee`, and `totalFees` values, potentially enabling up to 100% or higher fees on transfers.

Recommendation

We recommend setting an upper limit for `BTCBRewardsFee`, `liquidityFee`, `marketingFee`, and `totalFees` state variables.

Alleviation

[ShivaToken Team]: updated source code limits all fees to maximum value 20.

STS-04 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	Medium	ShivaToken.sol: 1602	Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[ShivaToken Team]: updated source code includes `nonReentrant` modifier on the function `transfer()`.

STS-05 | `deadWallet` Not Excluded From Dividends

Category	Severity	Location	Status
Logical Issue, Inconsistency	Minor	ShivaToken.sol: 1375	Resolved

Description

In the function `updateDividendTracker()`, the address `deadWallet` is not excluded from receiving dividends on the assignment of a `newDividendTracker`. The `deadWallet` address is excluded in the original `dividendTracker` during construction.

Recommendation

We recommend excluding the `deadWallet` address in the function `updateDividendTracker()` as follows:

```
1386  newDividendTracker.excludeFromDividends(deadWallet);
```

Alleviation

[ShivaToken Team]: updated source code removes function `updateDividendTracker()`.

STS-06 | `_uniswapV2Pair` Not Set As Automated Market Maker Pair

Category	Severity	Location	Status
Logical Issue, Inconsistency	Medium	ShivaToken.sol: 1398	Resolved

Description

In the function `updateUniswapV2Router()`, the resulting `_uniswapV2Pair` is not assigned as an automated market maker pair, as per construction.

Recommendation

We recommend including the call:

```
1399  _setAutomatedMarketMakerPair(_uniswapV2Pair, true);
```

Alleviation

[ShivaToken Team]: updated source code removes function `updateUniswapV2Router()`.

STS-07 | `_marketingWalletAddress` State Not Consistent on Change

Category	Severity	Location	Status
Logical Issue, Inconsistency	Minor	ShivaToken.sol: 1467	Resolved

Description

In the function `setMarketingWallet()`, the `_marketingWalletAddress` is not excluded from fees, anti-whale, or limit swap as per the constructor.

Recommendation

We recommend applying the same state on the new `_marketingWalletAddress` as assigned in the constructor:

```
excludeFromFees(_marketingWalletAddress, true);
_excludedFromAntiWhale[_marketingWalletAddress] = true;
_excludedLimitSwap[_marketingWalletAddress] = true;
```

Alleviation

[ShivaToken Team]: updated source code includes proper `_marketingWalletAddress` state changes upon a new address.

STS-08 | Usage of `transfer()` for sending BNB

Category	Severity	Location	Status
Volatile Code	Minor	ShivaToken.sol: 1712, 1854, 1856	Resolved

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs. This also includes BNB transfers on the Binance Smart Chain.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

Alleviation

[ShivaToken Team]: updated source code uses `sendValue()` for BNB transfers.

STS-09 | Gas Fee Passed to User

Category	Severity	Location	Status
Volatile Code, Control Flow	Minor	ShivaToken.sol: 1645~1651	Acknowledged

Description

In the function `_transfer()`, the functions `swapAndSendToFee`, `swapAndLiquify`, `swapAndSendDividends`, and `BuybackAndBurn` are called conditionally, mainly when the contracts SHIVA token balance exceeds `swapTokensAtAmount`. The execution of these functions help maintain intended tokenomics but pass the cost of gas to the address executing `_transfer()`.

Recommendation

We recommend reimbursing a disproportionately affected address for gas costs or delegating the logic to a separate function.

Alleviation

[ShivaToken Team]: acknowledged this finding.

STS-10 | Return Value Ignored

Category	Severity	Location	Status
Volatile Code	Minor	ShivaToken.sol: 1821, 1590	Resolved

Description

The linked functions invocations do not check the return value of the function call which should yield return values in case of a proper call.

Recommendation

We would advise to check the return value of the function and create appropriate response like usage of `require` with error message.

Alleviation

[ShivaToken Team]: updated source code includes proper checks for return values.

STS-11 | Requirement Always Passes

Category	Severity	Location	Status
Gas Optimization	Medium	ShivaToken.sol: 1426	Resolved

Description

In the function `updateMaxTransferAmountRate()`, the parameter `_maxTransferAmountRate` which is a `uint16` has maximum value of 2^16 or 65536. This constraint always satisfies the requirement:

```
1426  require(_maxTransferAmountRate <= 1000000, "SHIVA::updateMaxTransferAmountRate:
Max transfer amount rate must not exceed the maximum rate.");
```

Recommendation

We would like to confirm that the data type constraint for `maxTransferAmountRate` and `_maxTransferAmountRate` is intended to be `uint16`.

Alleviation

[ShivaToken Team]: updated source code changes the parameter of the function `updateMaxTransferAmountRate()` to a `uint256`.

STS-12 | Variable Declaration as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	ShivaToken.sol: 1229, 1070, 1234, 1237, 1238, 1252	Resolved

Description

Variables `BTCB`, `deadWallet`, `swapTokensAtAmount`, `BuyBackAtAmount`, `buybackFeeonSell` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend updating the contract or white paper to make them consistent with each other.

Alleviation

[ShivaToken Team]: updated source code declares `BTCB`, `deadWallet`, `swapTokensAtAmount`, `BuyBackAtAmount`, `buybackFeeonSell` as `constant`.

STS-13 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	ShivaToken.sol: 1459, 1463	Resolved

Description

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

- `setSelling()` which sets `selling`
- `setBuying()` which sets `buying`
- `setMarketingWallet()` which sets `_marketingWalletAddress`
- `setBTCBRewardsFee()` which sets `BTCBRewardsFee`
- `setLiquiditFee()` which sets `liquidityFee`
- `setMarketingFee()` which sets `totalFees`
- `blacklistAddress()` which sets `_isBlacklisted[account]`

Recommendation

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

Alleviation

[ShivaToken Team]: updated source code emits events during functions `setMarketingWallet()`, `setBTCBRewardsFee()`, `setLiquiditFee()`, `setMarketingFee()`, and `blacklistAddress()` and removes functions `setSelling()` and `setBuying()`.

STS-14 | Inconsistency With White Paper

Category	Severity	Location	Status
Inconsistency	Minor	ShivaToken.sol: 1237, 1643	Acknowledged

Description

The whitepaper for Shiva Token states that:

- "`After a certain amount of tokens have been stored in the contract (0.0001% of the total supply) it initiates a swap.`" However, the amount is hard coded as 20,000,000 tokens.
- "`The transaction limit is set to 0.1% of the total supply`" However, the `maxTransferAmountRate` is settable up to 6.5536% of total supply, currently 1%.

Recommendation

We recommend updating the contract or white paper to make them consistent with each other.

Alleviation

[ShivaToken Team]: acknowledged this finding.

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.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.