Security Assessment

WUSD & WUSDMaster

Aug 23rd, 2021

Table of Contents

Summary

This report has been prepared for Wault Finance to discover issues and vulnerabilities in the source code of the WUSD & WUSDMaster 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	WUSD & WUSDMaster
Platform	BSC
Language	Solidity
Codebase	https://github.com/WaultFinance/WUSD
Commit	• f66ab75eb7e995e99d56cc9f3c05c63b0ac2b22d • 8e6fd69a78c543a51659ad47ba254b53ad0609d7 • 5f50a2c7ffff7828c70299e8a9217cfbb926b8c1

Audit Summary

Delivery Date	Aug 23, 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	0	3	0
Medium	3	0	0	2	1	0
Minor	1	0	0	1	0	0
Informational	0	0	0	0	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
WUS	WUSD.sol	ef9fff70729004dc9aeb7d734b0378ada50e6949b883e1bac34dfe831a60575b
WUD	WUSDMaster.sol	c323c38eff278982c34b5264a495f62fe944b8fadc9e655f7fa0d06c24284a5a
WWW	WexWithdrawer.sol	7aa3597760bf47778c913699f3959ca2eb9476e7da70c56f6ef516e59061208b

Review Notes

Dependencies

There are a few depending injection contracts or addresses in the current project:

- `wusd`, `usdt`, `wex` and `wswapRouter` for the contract `WUSDMaster`;
- `wex` and `wusdMaster` for the contract `WexWithdrawer`.

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

Privilledged Functions

The `_minter` role in the contract `WUSD` can operate on the following functions:

- `WUSD.mint()` to mint WUSD to an account;
- `WUSD.burn()` to burn WUSD from an account.

The `_owner` role in the contract `WUSD` can operate on the following function:

- `Mintable.transferMintership()` to transfer the mintership to a new account.

The `_withdrawer` role in the contract `WUSDMaster` can operate on the following function:

- `WUSDMaster.withdrawWex()` to withdraw WEX to its account.

The `_owner` role in the contract `WUSDMaster` can operate on the following functions:

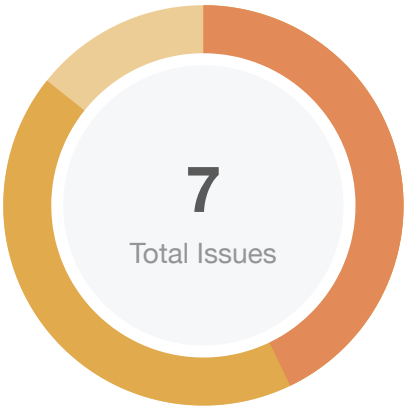
- `Withdrawable.transferWithdrawership()` to transfer withdrawership to arbitrary account;
- `WUSDMaster.pause()` to pause the contract;
- `WUSDMaster.unpause()` to unpause the contract;
- `WUSDMaster.setSwapPath()` to set a new swapping path for WEX-USDT exchange;
- `WUSDMaster.setWexPermille()` to set a new WEX permille;
- `WUSDMaster.setTreasuryPermille()` to set a new treasury permille;
- `WUSDMaster.setFeePermille()` to set a new fee permille;
- `WUSDMaster.setTreasuryAddress()` to set a new treasury address;
- `WUSDMaster.setStrategistAddress()` to set a new strategist address;
- `WUSDMaster.setMaxStakeAmount()` to set a new staking amount limitation;
- `WUSDMaster.setMaxRedeemAmount()` to set a new redemption amount limitation;
- `WUSDMaster.withdrawUsdt()` to withdraw USDT to the strategist account;
- `WUSDMaster.setMaxStakePerBlock()` to set a new staking amount limitation.

The `_owner` role in the contract `WexWithdrawer` can operate on the following functions:

- `WexWithdrawer.withdraw()` to withdraw WEX from `wusdMaster`;
- `WexWithdrawer.deposit()` to deposit WEX to `wusdMaster`;
- `WexWithdrawer.initiateMasterChange()` and `WexWithdrawer.changeMaster()` to change `wusdmaster`;
- `WexWithdrawer.cancelMasterChange()` to cancel the change of `wusdmaster`.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings



Critical	0 (0.00%)
Major	3 (42.86%)
Medium	3 (42.86%)
Minor	1 (14.29%)
Informational	0 (0.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
WUD-01	Potential WEX-USDT Exchange Rate Manipulation	Logical Issue	Major	Partially Resolved
WUD-02	Centralization Risk of Changing Settings and Withdrawing USDT	**Centralization / Privilege**	**Major**	Partially Resolved
WUD-03	Centralization Risk of Withdrawing WEX	**Centralization / Privilege**	**Medium**	Partially Resolved
WUD-04	Adequate USDT Balance is Not Guaranteed	Logical Issue	Medium	Acknowledged
WUS-01	Centralization Risk of Mint and Burn	**Centralization / Privilege**	**Major**	Partially Resolved
WWW-01	Centralization Risk of Withdrawing and Depositing WEX	**Centralization / Privilege**	**Medium**	Acknowledged
WWW-02	Centralization Risk of Changing `wusdMaster`	**Centralization / Privilege**	**Minor**	Acknowledged

WUD-01 | Potential WEX-USDT Exchange Rate Manipulation

Category	Severity	Location	Status
Logical Issue	Major	WUSDMaster.sol: 808, 859	Partially Resolved

Description

In the function `WUSDMaster.stake()`, a portion of staked USDT will be used to swap for WEX; in the function `WUSDMaster.claimUsdt()`, a portion of WEX will be used to swap for USDT. The WEX-USDT exchange rate might be manipulated by these processes.

For example, if WUSD is used for lending, people can make profits through the following strategy:

1. Borrow WUSD from the lending pool.
2. Call `WUSDMaster.redeem()` to get prepared for triggering `WUSDMaster.claimUsdt()` in the next block. Borrowed WUSD is sent to the contract in this step.
3. Call `WUSDMaster.claimUsdt()` and receive USDT. A portion of WEX in this contract is swapped for USDT so the price of WEX decreases in this step. Although the amount of claimed USDT is restricted in the previous step, people can create multiple accounts and trigger `WUSDMaster.claimUsdt()` for multiple accounts in one transaction so the change of WEX-USDT exchange rate should not be ignored.
4. Swap USDT for WEX at a low WEX price.
5. Call `WUSDMaster.stake()` to get prepared for claiming WUSD in the next block. USDT is sent to the contract, with a portion of it is used to swap for WEX, which leads to an increase in WEX's price.
6. Swap WEX for USDT. Considering the price of WEX has been increased by the previous step, people make profits by steps 4 and 6. Steps 3 to 6 should happen in one transaction.
7. Call `WUSDMaster.claimWusd()` and receive WUSD.
8. Return borrowed WUSD to the lending pool.

Recommendation

We advise the Wault Finance team to restrict the change of WEX-USDT exchange rate made by `WUSDMaster.stake()` and `WUSDMaster.claimUsdt()`.

Alleviation

The Wault Finance team introduced `lastBlockUsdtStaked` to restrict the amount of staked USDT in one block in the commit `5b635cdf3bac9f8c1d5187f90fb3eae7b831dc4a`.

With this restriction, the WEX-USDT exchange rate fluctuation in one block caused by `WUSDMaster.stake()` is limited, i.e. the step 5 in the described strategy is restricted. This restriction is not applied to `WUSDMaster.stake()`, so WEX price decrease caused by claiming USDT by WUSD holders is not limited. We assume this is acceptable business logic.

WUD-02 | Centralization Risk of Changing Settings and Withdrawing USDT

Category	Severity	Location	Status
Centralization / Privilege	**Major**	WUSDMaster.sol: 142, 748, 752, 756, 763, 770, 777, 784, 790, 796, 802, 909	Partially Resolved

Description

With the modifier `onlyOwner`, the `_owner` role has the authority to call the following sensitive functions to change the settings of the contract:

- `Withdrawable.transferWithdrawership()` to transfer withdrawership to arbitrary account;
- `WUSDMaster.pause()` to pause the contract;
- `WUSDMaster.unpause()` to unpause the contract;
- `WUSDMaster.setSwapPath()` to set a new swapping path for WEX-USDT exchange;
- `WUSDMaster.setWexPermille()` to set a new WEX permille;
- `WUSDMaster.setTreasuryPermille()` to set a new treasury permille;
- `WUSDMaster.setFeePermille()` to set a new fee permille;
- `WUSDMaster.setTreasuryAddress()` to set a new treasury address;
- `WUSDMaster.setStrategistAddress()` to set a new strategist address;
- `WUSDMaster.setMaxStakeAmount()` to set a new staking amount limitation;
- `WUSDMaster.setMaxRedeemAmount()` to set a new redemption amount limitation;
- `WUSDMaster.withdrawUsdt()` to withdraw USDT to the strategist account.

Any compromise to the `_owner` account may allow the hacker to manipulate the settings of the contract, withdraw staked USDT, or obtain the withdrawer role of the contract.

[**Update**]: The Wault Finance team added a new function `WUSDMaster.setMaxStakePerBlock()` to set a new staking amount limitation in one block in the commit `5f50a2c7ffff7828c70299e8a9217cfbb926b8c1`.

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 via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

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

- Timelock 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

[Wault]: The team acknowledged the issue, and the team used the timelock to delay the latency for 24H.

Timelock contract:

- 0x7a8d6c614635657660651db4802da08d17ddbbff

WUD-03 | Centralization Risk of Withdrawing WEX

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	WUSDMaster.sol: 916	Partially Resolved

Description

With the modifier `onlyWithdrawer`, the `_withdrawer` role has the authority to call the function `WUSDMaster.withdrawWex()` to withdraw WEX to its account.

Any missetting or compromise to the `_withdrawer` account may allow the hacker to withdraw WEX from the contract.

Recommendation

We advise the client to carefully manage the `_withdrawer` account's private key or set it to the correct contract address to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

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

- Timelock 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

[Wault]: The team acknowledged the issue, and the team used the timelock to delay the latency for 24H.

Timelock contract:

- 0x7a8d6c614635657660651db4802da08d17ddbbff

WUD-04 | Adequate USDT Balance is Not Guaranteed

Category	Severity	Location	Status
Logical Issue	Medium	WUSDMaster.sol: 870~871	Acknowledged

Description

The transactions in the aforementioned lines are supposed to transfer the specified amount of USDT to the treasury account and the user's account. However, considering the `_owner` can withdraw USDT from the contract to the strategist account, it is possible that these transactions got reverted because of the inadequate balance of USDT, hence leading to an undesirable scenario in which a user cannot fully redeem his/her collateral.

Recommendation

We advise the client to refactor the design of USDT withdrawal (e.g. limiting the amount of the withdrawal) so that a user's redemption can always be satisfied.

Alleviation

N/A

WUS-01 | Centralization Risk of Mint and Burn

Category	Severity	Location	Status
Centralization / Privilege	**Major**	WUSD.sol: 242, 597, 601	Partially Resolved

Description

With the modifier `onlyMinter`, the `_minter` role has the authority to call the following sensitive functions:

- `WUSD.mint()`: `_minter` allows to mint WUSD to arbitrary account;
- `WUSD.burn()`: `_minter` allows to burn WUSD from arbitrary account.

Meanwhile, the `_owner` role has the authority to call the function `Mintable.transferMintership()` to transfer the mintership to a new account.

If the `_minter` role is set to the `WUSDMaster` contract, the `WUSD.mint()` and `WUSD.burn()` behavior will follow the logic implemented in the `WUSDMaster` contract. However, if the `_minter` role is set to another account by mistake, or if the `_owner` account is hacked and set another `_minter` account, `WUSD.mint()` and `WUSD.burn()` might have unexpected behaviors leading to losses of users' assets.

Recommendation

We advise the client to carefully manage the `_minter` and `_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 are some feasible solutions that would also mitigate the potential risk:

- Timelock 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

[Wault]: The team acknowledged the issue, and the team used the timelock to delay the latency for 24H.

Timelock contract:

- 0x7a8d6c614635657660651db4802da08d17ddbbff

WWW-01 | Centralization Risk of Withdrawing and Depositing WEX

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	WexWithdrawer.sol: 508, 514	Acknowledged

Description

With the modifier `onlyOwner`, the `_owner` role has the authority to call the following sensitive functions to withdraw WEX from and deposit WEX to `wusdMaster`.

- `WexWithdrawer.withdraw()` to withdraw WEX from `wusdMaster` to the `WexWithdrawer` contract.
- `WexWithdrawer.deposit()` to transfer WEX from the `WexWithdrawer` contract to `wusdMaster`.

Any compromise to the `_owner` account may allow the hacker to drain WEX from `wusdMaster`.

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 via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

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

- Timelock 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

[**Wault Finance Team**]: The owner of the contract `WEXWithdrawer` will be connected to Oracle to control the peg.

WWW-02 | Centralization Risk of Changing `wusdMaster`

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	WexWithdrawer.sol: 520, 532, 542	Acknowledged

Description

With the modifier `onlyOwner`, the `_owner` role has the authority to call the following sensitive functions to change `wusdMaster`:

- `initiateMasterChange()` to initialize the change of `wusdMaster` with an arbitrary address and a delay (at least 48 hours).
- `cancelMasterChange()` to cancel the `wusdMaster` address ownership transfer immediately.
- `changeMaster()` to change the `wusdMaster` address after a delay since the process is initialized by `initiateMasterChange()`.

Any compromise to the `_owner` account may allow the hacker to change the `wusdMaster` 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 via smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

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

- 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

[**Wault Finance Team**]: The owner of the contract `WEXWithdrawer` will be connected to Oracle to control the peg.

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.

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.

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.