Code Security Assessment

Umbrella 2

Mar 21st, 2022

Table of Contents

Summary

This report has been prepared for Umbrella 2 to discover issues and vulnerabilities in the source code of the Umbrella 2 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	Umbrella 2
Description	Umbrella
Platform	Ethereum, BSC
Language	Solidity
Codebase	https://github.com/umbrella-network/overture-private
Commit	f9b9ee793cb2c385e1b2cea03b4825da4b371353

Audit Summary

Delivery Date	Mar 21, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

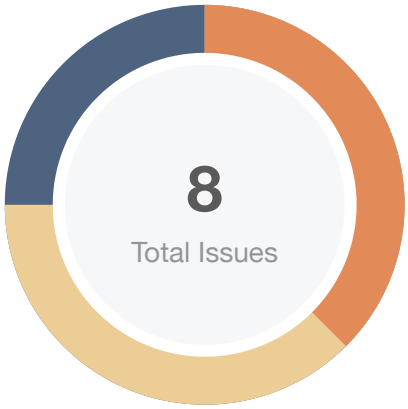
Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
BCK	interfaces/Blacklisted.sol	c00aedf32335736f7b4c3ed4d138eb5dfd2ad3805f80eca6253a19abed18274d
LSC	interfaces/LockSettings.sol	39b2da1bafe20b9ffdcd1bea5900c0320a0cb4223cd815c66aafab6dddde59162
ODT	interfaces/OnDemandTokenBridgable.sol	4762a798f6efc1310759e8f35961ec354416a6a321c91afce97aed0f2c57ac38
STV	interfaces/SwappableTokenV2.sol	70d9ef16efb869e9598a851331da7143b7974e5a33586c1dd5fd1ba621f2d8e6
UMB	interfaces/rUMBV2.sol	b225d9339c2f6e95034b342c0e1436d8a81b5151a5c18c38a7e478b9827f71f7
SLC	staking/StakingLens.sol	f0648a991c56363086835bba43dbcc1a0baceec399a9e0e26585e0ba2e048c13
SLK	staking/StakingLockable.sol	e9f39dc6f4333a534469701424e1d15d3571017348eecd3cde111018b547e120
SRV	staking/StakingRewardsV2.sol	e455901d7ae77ae1bace3595d0d975de0ae135d1bdaa436ca3decfb9bb7af3ec
UMC	rUMB2.sol	2b8807e44878344d052670d72ee0c46468abf5a67144927e6148b59095b9e189
UMS	rUMB2BSC.sol	bce425f0eb618636d6953726d448d9e76761f76341cf6be39882daef2b29e307

Findings



Critical	**0** (0.00%)
Major	**3** (37.50%)
Medium	**0** (0.00%)
Minor	**3** (37.50%)
Informational	**2** (25.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
CKP-01	SafeMath Not Used	Mathematical Operations	Minor	Acknowledged
ODT-01	Logical issue of the function `_beforeTokenTransfer()`	**Logical Issue, Centralization / Privilege**	**Major**	Mitigated
SLK-01	Improper Usage of `public` and `external` Type	Gas Optimization	Informational	Acknowledged
SLK-02	Logical issue of the bonus for lock	Logical Issue	Informational	Acknowledged
SLK-03	Logical issue of the function `_getReward()`	Logical Issue	Minor	Acknowledged
STV-01	Logical issue of function `swapForUMB()`	Logical Issue	Minor	Acknowledged
STV-02	Invalid daily limit	Logical Issue	Major	Acknowledged

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

In the contract `Blacklisted` the role `owner` has authority over the following functions:

- `setupBlacklist()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

In the contract `LockSettings` the role `owner` has authority over the following functions:

- `removePeriods()`
- `setLockingTokenSettings()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

In the contract `OnDemandToken` the role `owner` has authority over the following functions:

- `setupMinter()`
- `setupMinters()`

Any compromise to the `owner/minter` account may allow a hacker to take advantage of this authority.

In the contract `OnDemandTokenBridgable` the role `owner` has authority over the following functions:

- `setupBridge()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

In the contract `SwappableTokenV2` the role `owner` has authority over the following functions:

- `startEarlySwap()`
- `setDailyCup()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

In the contract `rUMB2/rUMB2BSC` the role `owner/minter` has authority over the following functions:

- `mint()`

Any compromise to the `owner/minter` account may allow a hacker to take advantage of this authority.

In the contract `StakingLockable` the role `owner` has authority over the following functions:

- `finishFarming()`

Any compromise to the `owner` account may allow a hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND

- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team acknowledged this issue and they will transfer the ownership to the multi-signature wallet in their own timeframe.

CKP-01 | SafeMath Not Used

Category	Severity	Location	Status
Mathematical Operations	Minor	interfaces/SwappableTokenV2.sol staking/StakingLockable.sol	Acknowledged

Description

SafeMath from OpenZeppelin is not used in the following functions which makes them possible for overflow/underflow and will lead to an inaccurate calculation result.

- `SwappableTokenV2().swapForUMB()`
- `StakingLockable.notifyRewardAmount()`
- `StakingLockable.rewardPerToken()`
- `StakingLockable.earned()`
- `StakingLockable._stake()`
- `StakingLockable._addLock()`
- `StakingLockable._withdrawUnlockedTokens()`

Recommendation

We advise the client to use OpenZeppelin's SafeMath library for all of the mathematical operations.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/SafeMath.sol

Alleviation

The team acknowledged this issue and they stated:

"They operated only based on amounts from 3 known tokens.They need only 89bits to store total supply. The max multiplier is 5e6 (5x) when they lock tokens."

ODT-01 | Logical Issue Of The Function `_beforeTokenTransfer()`

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	interfaces/OnDemandTokenBridgable.sol: 21	Mitigated

Description

According to the following codes, the function `_beforeTokenTransfer()` will check the `from` and `to` addresses and calls the function `_mint()` to mint tokens when the balance is not enough.

```
    function _beforeTokenTransfer(address _from, address _to, uint256 _amount) internal
virtual override {
        if (_from != address(0) && _to != address(0) && bridges[msg.sender]) {
            uint256 balance = balanceOf(msg.sender);

            if (balance < _amount) {
                _mint(msg.sender, _amount - balance);
            }
        }
    }
```

The `OnDemandTokenBridgable._beforeTokenTransfer()` will be called in the `rUMB2._beforeTokenTransfer()` and rUMB2 has a total supply limit. The function `_mint()` doesn't check if the total supply limit exceeds.

As a result, the total supply may exceed the total supply limit.

Any compromise to the `bridges` account may allow a hacker to take advantage of this authority to mint the tokens without limit.

Recommendation

We recommend adding reasonable checks to avoid this logical issue.

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The team heeded our advice and they added the check of the totalsupply to avoid exceeding the total supply limit in the commit `41f9363413d9d99dabfc4cb18c69d970d7b0cdbf`.

SLK-01 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	staking/StakingLockable.sol: 270, 274	Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

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

SLK-02 | Logical Issue Of The Bonus For Lock

Category	Severity	Location	Status
Logical Issue	Informational	staking/StakingLockable.sol	Acknowledged

Description

According to the following codes, users can choose to lock their tokens for a period to get additional rewards. This additional rewards is calculated through `balances[_user].lockedWithBonus`.

```
    function _lockTokens(address _user, address _token, uint256 _amount, uint256 _period)
internal notPaused {
        uint256 multiplier = multipliers[_token][_period];
        require(multiplier != 0, "invalid period or not supported token");

        uint256 stakeBonus = calculateBonus(_amount, multiplier);

        _stake(_token, _user, _amount, stakeBonus);
        _addLock(_user, _token, _amount, _period, multiplier);
    }
```

After the locked period, users can get their token back. However, users may choose not to unlock the token in time, at which point their staked tokens is no different from a common stake, but any enjoy additional rewards.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team acknowledged this issue and they stated:

"The current implementation aligns with the original project design."

SLK-03 | Logical Issue Of The Function `_getReward()`

Category	Severity	Location	Status
Logical Issue	Minor	staking/StakingLockable.sol: 467	Acknowledged

Description

According to the following codes, the rewards are distributed to users through minting.

```
function _getReward(address _user, address _recipient)
    internal
    nonReentrant
    updateReward(_user)
    returns (uint256 reward)
{
    reward = balances[_user].rewards;

    if (reward != 0) {
        balances[_user].rewards = 0;
        OnDemandToken(address(rUmb2)).mint(_recipient, reward);
        emit RewardPaid(_user, reward);
    }
}
```

The `rUmb2` token has a max mint limit. As a result, the reward distribution may fail due to this limit.

Recommendation

We recommend stating for this.

Alleviation

[Certik Response]:

In commit `bf23045187781694d61834597f0c1055c5aa0e1a`, the variable `maxEverTotalRewards` is added and used to check whether the `totalRewardsSupply` is over the max mint limit of rUMB2. The `totalRewardsSupply` is the total rewards minted to the users. Both the functions `_getReward()`, `swap1to2()`, `OnDemandTokenBridgable._beforeTokenTransfer()` will call the function `rUMB2.mint()`. These calls will increase the total supply of rUMB2, which has the max mint limit.

For example, if the max mint limit of rUMB2 is 1000e18, and the `totalRewardsSupply` is 900e18. The result of the checking `totalRewardsSupply <= maxEverTotalRewards` is true.

However, the functions `swap1to2()` and `OnDemandTokenBridgable._beforeTokenTransfer()` are called and the amount of minted is 200e18. As a result, the last 100e18 rewards in the function `_getReward()` will be fail to mint.

[Umbrella Team]:

In general, what you saying is true but only if we internally miscalculate rewards we want to distribute and we “say” to contract it should distribute more that we have, then yes minting will fail - and that’s good of course because this is after all protection to not have more than MAX total supply. In reality, the MAX supply for rUMB2 already covers the supply for rUMB1 (because of the swap1to2), we even have some buffer in case of emergency. So 200M tokens is a sum of all rUMBs + buffer.

STV-01 | Logical Issue Of Function `swapForUMB()`

Category	Severity	Location	Status
Logical Issue	Minor	interfaces/SwappableTokenV2.sol: 48	Acknowledged

Description

According to the following codes, the function `swapForUMB()` is used to swap the user's all rUMB2 tokens to be UMB tokens.

```
    function swapForUMB() external {
        SwapData memory data = swapData;

        (uint256 limit, bool freshLimit) = _currentLimit(data);
        require(limit != 0, "swapping period not started OR limit");

        uint256 amountToSwap = balanceOf(msg.sender);
        require(amountToSwap != 0, "you dont have tokens to swap");

        uint32 amountWoDecimals = uint32(amountToSwap / ONE);
        require(amountWoDecimals <= limit, "daily CUP limit");

        swapData.usedLimit = uint32(freshLimit ? amountWoDecimals : data.usedLimit +
amountWoDecimals);
        swapData.swappedSoFar += amountWoDecimals;
        if (freshLimit) swapData.dailyCupTimestamp = uint32(block.timestamp);

        _burn(msg.sender, amountToSwap);
        umb.swapMint(msg.sender, amountToSwap);

        emit LogSwap(msg.sender, amountToSwap);
    }
```

However, there exists a daily cap for the swap amount. As a result, the user can't swap when the balance is over the daily cap.

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team acknowledged this issue and they stated:

"The current implementation aligns with the original project design."

STV-02 | Invalid Daily Limit

Category	Severity	Location	Status
Logical Issue	Major	interfaces/SwappableTokenV2.sol	Acknowledged

Description

According to the following codes, the user's balance drops divided by the constant variable `ONE(1e18)` and compared with the daily limit.

```
        (uint256 limit, bool freshLimit) = _currentLimit(data);
        require(limit != 0, "swapping period not started OR limit");

        uint256 amountToSwap = balanceOf(msg.sender);
        require(amountToSwap != 0, "you dont have tokens to swap");
        uint32 amountWoDecimals = uint32(amountToSwap / ONE);
        require(amountWoDecimals <= limit, "daily CUP limit");

        swapData.usedLimit = uint32(freshLimit ? amountWoDecimals : data.usedLimit +
amountWoDecimals);
        swapData.swappedSoFar += amountWoDecimals;
```

In solidity, the division truncation problem exists for division operations. Hence, when the user's balance is below 1e18, like 0.9e18. The `usedLimit` and `swappedSoFar` will record incorrectly. As a result, the user can swap 0.9e18token for many times. The actual number of tokens swapped can easily exceed the daily limit.

Recommendation

We recommend calculating the usedLimit correctly.

Alleviation

The team acknowledged this issue and they stated:

"They did huge gas optimization but there is small flaw exactly as described, where user can bypass limit, but this is only for balance less than one token, the transaction will cost more than swap is worth so nobody will do it otherwise they lose money."

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.