CERTIK

Security Assessment

Umbrella Network2023

CertiK Verified on Mar 22nd, 2023

CertiK Verified on Mar 22nd, 2023

Umbrella Network2023

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 03/22/2023	N/A

CODEBASE

https://github.com/umbrella-network/overture-private

ETH Deployments:

0x2d9D79B3189377449aB2AA4bBD2cd2651e0b85BE

...View All

COMMITS

Base: b726b660c537ec13766b6dc788db68e0b0b74211

Update1: 73e0f876e814d0d03f44d53e49b2bb9f4d8d5a4e

Update2: d7ad20d5018bb914a8de71da08e8ff5327924dc5

...View All

Vulnerability Summary

Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved
7	7	0	0	0	0	0

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.
3	Minor	3 Resolved	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.
4	Informational	4 Resolved	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 | UMBRELLA NETWORK2023

CODEBASE | UMBRELLA NETWORK2023

Repository

https://github.com/umbrella-network/overture-private

ETH Deployments:
0x2d9D79B3189377449aB2AA4bBD2cd2651e0b85BE
0xB67D91E38fbA6CfCb693d3f4598F8bd1e6e68AE3

BSC Deployments:
0x55881395d209397b0c00bCeBd88abC1386f7aBe7
0x6Ff6B943D20B611E81a581c1E7951A6Dc0AC3455

Commit

Base: b726b660c537ec13766b6dc788db68e0b0b74211
Update1: 73e0f876e814d0d03f44d53e49b2bb9f4d8d5a4e
Update2: d7ad20d5018bb914a8de71da08e8ff5327924dc5
Update3: b0a642b2faafaddae15fef944ddbae1388184bab
Update4: ca90201f099dc78b0925f0af16907a44c32e524c
Update5: 31879b3b767860d520ca32693d91720c53690875

AUDIT SCOPE | UMBRELLA NETWORK2023

1 file audited ● 1 file with Resolved findings

ID	Repo	Commit	File	SHA256 Checksum
● SRU	umbrella-network/overture-private	b726b66	contracts/staking/StakingRewards.sol	3cb83a267e01924cc87884ee8087b99b1db45f80571e7c88c2079dcfce1fe9ec

APPROACH & METHODS | UMBRELLA NETWORK2023

This report has been prepared for Umbrella to discover issues and vulnerabilities in the source code of the Umbrella Network2023 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:

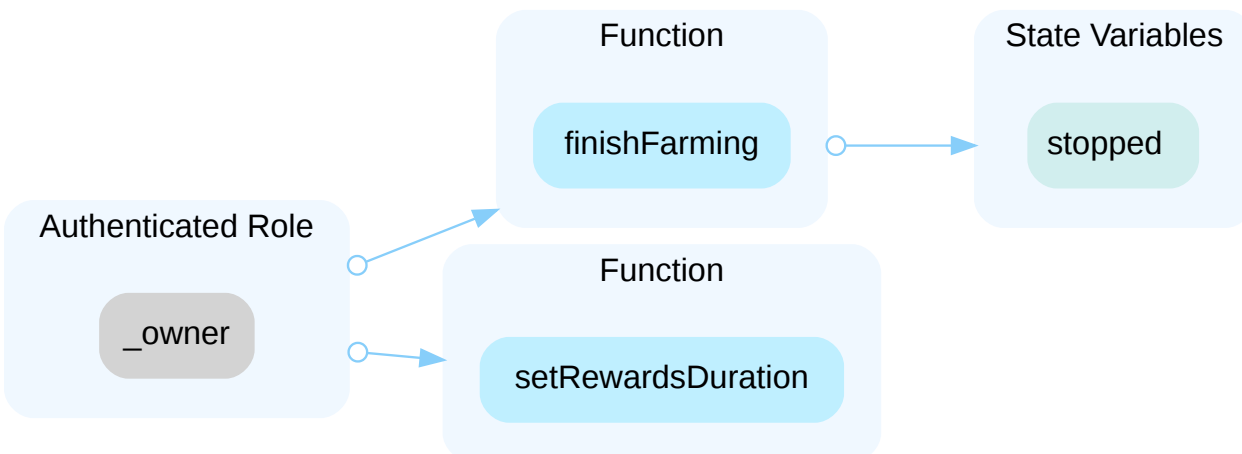
- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

DECENTRALIZATION EFFORTS | UMBRELLA NETWORK2023

Description

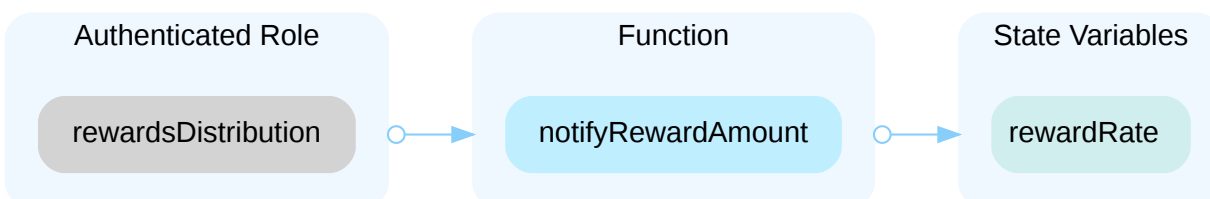
In the contract `StakingRewards` the role `onlyOwner` has authority over the functions shown in the diagram below. Any compromise to the `onlyOwner` account may allow the hacker to take advantage of this by:

- Calling `finishFarming()`, which will finalize a stopping point for staking.
    - Which will set the following functions that uses `whenActive` to fail:
    - `notifyRewardAmount()`
    - `setRewardsDuration()`
    - `finishFarming()`
- Calling `setRewardsDuration()` which will set the `rewardsDuration` to a new value which may increase or decrease rewards.



In the contract `StakingRewards` the role `rewardsDistribution` has authority over the functions shown in the diagram below. Any compromise to the `rewardsDistribution` account may allow the hacker to take advantage of this by:

- Allowing new `rewardRate` to enter for the remaining portion or for the new duration.



- The centralized entity is responsible for ensuring the reward token are always available for users to receive their staked rewards.

Recommendations

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., multisignature 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.

Alleviation

Both the `owner` role and `rewardsDistribution` were set to the following multi-signs which both require 3 out of 4 signatures:

- ETH: 0xa800863644ce2F0e6c7Bb3226E40Fda8ec649881
- BSC: 0x28e2fd9620ae1ce68dde1e246f352a3925c65433

FINDINGS | UMBRELLA NETWORK2023

7	0	0	0	3	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Umbrella Network2023. Through this audit, we have uncovered 7 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
SRH-07	`notifyRewardAmount()` Will Leave Dust In Contract	Logical Issue	Minor	Resolved
SRU-02	Users Can Potentially Stake Past Closed	Logical Issue	Minor	Resolved
SRU-03	Possible Overflow/Underflow	Mathematical Operations	Minor	Resolved
SRU-04	Missing Zero Address Validation	Volatile Code	Informational	Resolved
SRU-05	Incompatibility With Deflationary Tokens	Logical Issue	Informational	Resolved
SRU-06	Out Of Scope Dependency	Volatile Code	Informational	Resolved
SRU-08	External View Functions Can Be Separated	Coding Style	Informational	Resolved

SRH-07 | `notifyRewardAmount()` WILL LEAVE DUST IN CONTRACT

Category	Severity	Location	Status
Logical Issue	Minor	contracts/staking/StakingRewards.sol (Update1): 64	Resolved

Description

When `notifyRewardAmount()` is called the first time `timeData.periodFinish` will be set to the current `block.timestamp + t.rewardsDuration`. With the current value of `rewardsDuration` being 2592000 seconds, this should run for 30 days after being called. If `stake()` is not called in the same block as `notifyRewardAmount()` this will cause a small amount of tokens to stay inside the contract. When a user calls `stake()` for the first time, it will call the modifier `updateReward()`. This is an issue due to `timeData.lastUpdateTime` being equal to `lastTimeRewardApplicable` which does not take in the time before the first call of `stake()`.

Scenario

The contract has 5184000 tokens sent to cover all rewards.

1. Bob is the deployer and calls `notifyRewardAmount()` and the `rewardRate` will be 2. The total reward will be (2 * 2592000).
2. Alice decides to be the first user to stake. Alice stakes 1 hour after `notifyRewardAmount()` is called.
3. Because the gap in time between `stake()` and `notifyRewardAmount()`, this will cause (3600*2) tokens to be inside the contract and cannot be handed out.

Recommendation

We recommend factoring in this delay of time to ensure tokens are not locked into the contract.

Alleviation

`[CertiK]`: The client fixed the issues in the following commit: 31879b3b767860d520ca32693d91720c53690875.

SRU-02 | USERS CAN POTENTIALLY STAKE PAST CLOSED

Category	Severity	Location	Status
Logical Issue	Minor	contracts/staking/StakingRewards.sol (Base): 226	Resolved

Description

The modifier `whenActive` is not attached to staking. After farming is finished, users can still enter and exit the staking pool.

Recommendation

We recommend adding `whenActive` to `_stake()`. If there are not scenarios where staking should be paused and farming is active, then `notPaused()` can be removed.

Alleviation

`[CertiK]`: The team acknowledged the finding but stated this was by design. The modifier `onlyActive` is for the owner to protect against restarting an old contract.

SRU-03 | POSSIBLE OVERFLOW/UNDERFLOW

Category	Severity	Location	Status
Mathematical Operations	Minor	contracts/staking/StakingRewards.sol (Base): 105~106	Resolved

Description

Since `StakingRewards` is less than solidity ^0.8.0, overflow and underflows checking is not built in. This makes it possible for overflow/underflow to occur and lead to inaccurate calculations.

One example of a possible overflow is:

- Deployer calls `notifyRewardAmount()` then immediately calls it again with a value greater than `(2**256-1) - (remaining * rewardRate)` causing `newRewardRate` to overflow and potentially causing rewards to be lower than intended.

```
105  uint256 leftover = remaining * rewardRate;
106  newRewardRate = (_reward + leftover) / t.rewardsDuration;
```

Proof of Concept

The following test case written with Foundry will demonstrate that overflow is possible:

1. Deploy the contract and call `notifyRewardAmount()` with a `_reward` amount of 10,000,000.
2. Skip forward in time by 15555 seconds
3. Call `notifyRewardAmount()` again with the `_reward` amount of (2 ** 256) - 1;
4. This will cause the sum of `leftover` and `_reward` to overflow as it will exceed the maximum value of a uint256. This will cause the `newRewardRate` to be a lower `rewardRate` than expected.

Note that the check that `totalRewardsSupply <= maxEverTotalRewards` will be passed due to another overflow. This happens because the `totalRewardsSupply` is the sum `timeData.totalRewardsSupply + _reward` which will be `10_000_000 + ((2 ** 256) - 1)` that will overflow to `9_999_999`.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.7.5;
pragma abicoder v2;
import "forge-std/Test.sol";
import "../src/overture-private-develop/contracts/staking/StakingRewards.sol";
import "../src/overture-private-develop/contracts/UMB.sol";
import "../src/overture-private-develop/contracts/rUMB2.sol";

contract testOverflow is Test {
    rUMB2 testRUMB;
    UMB testUMB;
    StakingRewards StakingTest;
    address deployer = address(0xACDE);
    uint256 public MAX_UINT256 = 2 ** 256 - 1;

    function setUp() public {
        vm.startPrank(deployer);
        testUMB = new UMB(
            deployer,
            deployer,
            1000000000000,
            (1000000000000 * 10),
            "UMB",
            "UMB"
        );
        testRUMB = new rUMB2(
            deployer,
            1000000000000, //uint256 _maxAllowedTotalSupply,
            1000000,
            10000000,
            "rUMB1",
            "rUMB1",
            address(testUMB)
        );

        StakingTest = new StakingRewards(
            deployer,
            deployer,
            address(testRUMB),
            address(testRUMB)
        );
        vm.stopPrank();
    }

    //steps to setup in StakingRewards.sol --> set leftover to a public variable to
look at the value change.
    //not necessary unless looking at the leftover value
```

```
    //The calculation of leftover should be 3 * (2592000 - 15555) = 7729335
    //The calculation of newRewardRate = ((2**256 -1) + 7729335) / 2592000
    //The numerator exceeds the maximum uint256 and will overflow so the calculation
becomes 7729334/2592000
    //Which equals 2.98 which is truncated to 2
    //This can cause rewards to be lower than intended.

    function testOverFlow() public {
        vm.startPrank(deployer);
        StakingTest.notifyRewardAmount(10000000); //input is arbitray and set to
10,000,000
        emit log_named_uint("Reward Rate", StakingTest.rewardRate());
        skip(15555); //skip forward in time by 15555 seconds
        StakingTest.notifyRewardAmount(
            MAX_UINT256 // call again but with a new value of uint256 max value
        );
        emit log_named_uint("Leftover after Overflow", StakingTest.leftover());
        emit log_named_uint("Reward Rate", StakingTest.rewardRate());
    }
}
```

Recommendation

We recommend using OpenZeppelin's SafeMath library for all relevant mathematical operations.

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

Alleviation

`[CertiK]` : The client fixed the issues in the following commit: ca90201f099dc78b0925f0af16907a44c32e524c.

SRU-04 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Informational	contracts/staking/StakingRewards.sol (Base): 79~81, 85~87	Resolved

Description

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

```
82        stakingToken = IERC20(_stakingToken);
83        rewardsToken = IERC20(_rewardsToken);
84        rewardsDistribution = _rewardsDistribution;
```

- `_rewardsDistribution`, `_rewardsToken`, and `_stakingToken` are not zero-checked before being used.

Recommendation

We recommend adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

`[CertiK]`: The client fixed the issues in the following commit: 73e0f876e814d0d03f44d53e49b2bb9f4d8d5a4e.

SRU-05 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	Informational	contracts/staking/StakingRewards.sol (Base): 236~237, 243, 259, 261, 264	Resolved

Description

When transferring deflationary ERC20 tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user sends 100 deflationary tokens (with a 10% transaction fee) via `stake()`, only 90 tokens actually arrived to the contract. However, a failure to discount such fees may allow the same user to call `withdraw()` or `exit()` some or all of their tokens from the contract, which causes the contract to lose incorrectly manage the amount of tokens in a users balance. This is a problem as not all token holders will not be able to withdraw their asset due to not storing the correct value. In turn, this allows some users to withdraw more than they should and leave other users with nothing.

Recommendation

We recommend regulating the set of tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

`[CertiK]` : The client stated the following token will be used for the protocol and it is not deflationary:

ETH: 0x6fC13EACE26590B80cCCAB1ba5d51890577D83B2.

BSC: 0x846F52020749715F02AEf25b5d1d65e48945649D.

SRU-06 | OUT OF SCOPE DEPENDENCY

Category	Severity	Location	Status
Volatile Code	Informational	contracts/staking/StakingRewards.sol (Base): 79~81, 85~87	Resolved

Description

The contract serves as the underlying entity to interact with multiple external contracts such as `_rewardsDistribution`, `_stakingToken`, and `_rewardsToken`. The scope of the audit treats out of scope contracts as black boxes, assumes their functional correctness, and the audited contracts interact with those contracts in a correct way. However, in the real world, those contracts might contain logic issues or security vulnerabilities, and this may lead to lost or stolen assets.

Recommendation

We understand that the business logic requires interaction with `_rewardsDistribution`, `_stakingToken`, and `_rewardsToken`. We encourage the team to ensure the correctness and security of out-of-scope contracts to prevent unexpected errors from happening.

Alleviation

`[Umbrella Network]`: UMB will be used as reward and staking tokens:

- ETH: 0x6fC13EACE26590B80cCCAB1ba5d51890577D83B2.
- BSC: 0x846F52020749715F02AEf25b5d1d65e48945649D.

We will also use these for staking tokens:

- ETH: 0xB1BbeEa2dA2905E6B0A30203aEf55c399C53D042 (Uniswap v2 LP token).
- BSC: 0xFfD8eEFb9F0Ba3C60282fd3E6567A2C78C994266 (PancakeSwap LP token).

Distributor: Umbrella Gnosis multisig

- ETH: 0xa800863644ce2F0e6c7Bb3226E40Fda8ec649881.
- BSC: 0x28E2Fd9620AE1ce68Dde1e246F352A3925C65433.

UMB token contract was audited. You can find the audit report here: https://drive.google.com/file/d/1hvfhXMhGxAH_ztvtmJXucwlrzHocrviC/view?usp=sharing. It also has been battle-tested as it

is being used on Mainnet for over 2 years. The LP token contracts have both been battle-tested.

SRU-08 | EXTERNAL VIEW FUNCTIONS CAN BE SEPARATED

Category	Severity	Location	Status
Coding Style	Informational	contracts/staking/StakingRewards.sol (Base): 167~193	Resolved

Description

The contract `StakingRewards` has the following sections for restrictive and mutative functions. However, the view functions should be put into the `view` category to follow the same standard and be consistent.

Recommendation

We recommend changing this to stay consistent with the rest of the contract.

Alleviation

`[CertiK]` : The client fixed the issues in the following commit: d7ad20d5018bb914a8de71da08e8ff5327924dc5.

OPTIMIZATIONS | UMBRELLA NETWORK2023

ID	Title	Category	Severity	Status
SRU-09	Unnecessary Initilization	Gas Optimization	Optimization	Resolved

SRU-09 | UNNECESSARY INITILIZATION

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/staking/StakingRewards.sol (Base): 34	Resolved

Description

The variable `rewardRate` will be initialized to `0` by default.

Recommendation

We recommend not initializing this variable directly to save gas.

Alleviation

`[CertiK]` : The client fixed the issues in the following commit: b0a642b2faafaddae15fef944ddbae1388184bab.

APPENDIX | UMBRELLA NETWORK2023

Finding Categories

Categories	Description
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.
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.

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.