CERTIK

Security Assessment

HydroLink

Nov 15th, 2021

Table of Contents

Summary

This report has been prepared for HydroLink to discover issues and vulnerabilities in the source code of the HydroLink 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	HydroLink
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x5523636a2ae324000eb0f239c474bb04346fec31#code
Commit	

Audit Summary

Delivery Date	Nov 15, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

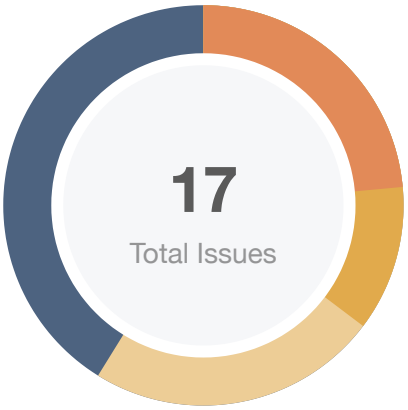
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	4	0	0	3	1	0
Medium	2	0	0	2	0	0
Minor	4	0	0	4	0	0
Informational	7	0	0	7	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
HLH	HydroLink.sol	61630c298dc3659bacf32113935538124d6891dbce6a8e18e557b308359aa34e

Findings



Critical	**0** (0.00%)
Major	**4** (23.53%)
Medium	**2** (11.76%)
Minor	**4** (23.53%)
Informational	**7** (41.18%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
HLH-01	Unlocked Compiler Version	Language Specific	Informational	Acknowledged
HLH-02	Possible To Gain Ownership After Renouncing The Contract Ownership	**Logical Issue, Centralization / Privilege**	**Major**	Acknowledged
HLH-03	Typos In The Contract	Coding Style	Informational	Acknowledged
HLH-04	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
HLH-05	Third Party Dependencies	Control Flow	Minor	Acknowledged
HLH-06	Lack of Zero Address Validation	Volatile Code	Minor	Acknowledged
HLH-07	The Purpose Of Function `deliver`	Control Flow	Informational	Acknowledged
HLH-08	Incorrect Error Message	Logical Issue	Minor	Acknowledged
HLH-09	Missing Emit Events	Gas Optimization	Informational	Acknowledged
HLH-10	Return Value Not Handled	Volatile Code	Informational	Acknowledged
HLH-11	Centralized Risk In `addLiquidity`	**Centralization / Privilege**	**Major**	Partially Resolved
HLH-12	Redundant Code	Logical Issue	Informational	Acknowledged
HLH-13	Lack of Input Validation	Volatile Code	Minor	Acknowledged
HLH-14	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged

ID	Title	Category	Severity	Status
HLH-15	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
HLH-16	Not Update `rOwned`	Logical Issue	Medium	Acknowledged
HLH-17	Potential Sandwich Attacks	Logical Issue	Medium	Acknowledged

HLH-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 7	Acknowledged

Description

The contract contains unlocked compiler versions. An unlocked compiler version in the contract's source code permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is a general practice to alternatively lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and in doing so be able to identify emerging ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-02 | Possible To Gain Ownership After Renouncing The Contract Ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	projects/HydroLink/contracts/HydroLink.sol (1230346): 392	Acknowledged

Description

An owner has the possibility to gain ownership of the contract even if he calls function `renounceOwnership` to renounce the ownership. This can be achieved by performing the following operations:

1. Call `lock` to lock the contract. The variable `_previousOwner` is set to the current owner.
2. Call `unlock` to unlock the contract.
3. Call `renounceOwnership` to leave the contract without an owner.
4. Call `unlock` to regain ownership.

Recommendation

We advise updating/removing `lock` and `unlock` functions in the contract; or removing the `renounceOwnership` if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound finance as reference. Reference: https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol

Alleviation

`[HydroLink]`: The HydroLink project is a centralized project. The contract ownership will be maintained for the foreseeable future and as such the renounce0wnership function will not be called. Consequently, there is no reason for the operations stated to be used to regain ownership.

HLH-03 | Typos In The Contract

Category	Severity	Location	Status
Coding Style	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 437, 711, 903	Acknowledged

Description

There are several typos in the code and comments.

1. In the following code snippet, `tokensIntoLiqudity` should be `tokensIntoLiquidity`.

```
event SwapAndLiquify(
    uint256 tokensSwapped,
    uint256 ethReceived,
    uint256 tokensIntoLiqudity
);
```

2. `recieve` should be `receive` and `swaping` should be `swapping` in the line of comment `//to recieve ETH from uniswapV2Router when swaping`.
3. `geUnlockTime()` should be `getUnlockTime()`.

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-04 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	Major	projects/HydroLink/contracts/HydroLink.sol (1230346): 737	Acknowledged

Description

`_rTotal` tokens were sent to the `tokenOwner` when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process.

Alleviation

`[HydroLink]`: There is no risk of future token distribution. HydroLink acknowledges the importance of transparent communication of material business activities, including retrospectively, and consequently will include a clear explanation of the initial token distribution a process on our ‘transparency statement’ available at hydrolink.app.

HLH-05 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	Minor	projects/HydroLink/contracts/HydroLink.sol (1230346): 745	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party DEX. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised and lead to assets being lost or stolen.

Recommendation

We understand that the business logic of the HydroLink protocol requires the interaction DEX for adding liquidity to the HydroLink-BNB pool and swap tokens. We encourage the team to constantly monitor the status of those 3rd parties to mitigate negative outcomes when unexpected activities are observed.

Alleviation

`[HydroLink]`: In accordance with this, the team will closely monitor PancakeSwap (and any future third party dependency) for unexpected outcomes.

HLH-06 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	projects/HydroLink/contracts/HydroLink.sol (1230346): 723	Acknowledged

Description

The variables `routerAddress` and `tokenOwner` should be verified as non-zero values to prevent being mistakenly assigned as `address(0)` in the `constructor()` function.

Recommendation

We advise the client to check that the addresses are not zero in `constructor()` like as follows:

```
require(routerAddress != address(0), "HydroLink: routerAddress is a zero address");
```

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-07 | The Purpose Of Function `deliver`

Category	Severity	Location	Status
Control Flow	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 815	Acknowledged

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the HydroLink token balance of the caller by `rAmount`, which is `tAmount` reduces the transaction fee. Then, the function adds `tAmount` to variable `_tFeeTotal`, which represents the contract's total transaction fee. We wish the team could explain more on the purpose of having such functionality.

Alleviation

`[HydroLink]`: The developers had tested the functionality in many different environments. HydroLink accepts the code is unnecessary but also categorizes the existence of the function as redundant and consequently not a material risk. Further, given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-08 | Incorrect Error Message

Category	Severity	Location	Status
Logical Issue	Minor	projects/HydroLink/contracts/HydroLink.sol (1230346): 852	Acknowledged

Description

The error message in `require(_isExcluded[account], "Account is already excluded")` does not describe the error correctly.

Recommendation

The message "Account is already excluded" can be changed to "Account is not excluded" .

Alleviation

`[HydroLink]`: HydroLink has confirmed this does not affect the mathematical integrity of contract functionality. As such, HydroLink proposes issue resolution in an explanatory sense - a statement informing the community within our ‘transparency statement’ (accessible at hydrolink.app) which will directly address the error message.

HLH-09 | Missing Emit Events

Category	Severity	Location	Status
Gas Optimization	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 874, 878, 882, 886, 890, 894	Acknowledged

Description

Functions that affect the status of sensitive variables should be able to emit events as notifications to customers.

- `excludeFromFee()`
- `includeInFee()`
- `setTaxFeePercent()`
- `setLiquidityFeePercent()`
- `setNumTokensSellToAddToLiquidity()`
- `setMaxTxPercent()`

Recommendation

We advise the client to add events for sensitive actions and emit them in the function as follows:

```
event TaxFeePercentUpdated(uint256 oldTaxFee, uint256 taxFee);

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
   emit TaxFeePercentUpdated(_taxFee,taxFee);
   _taxFee = taxFee;
}
```

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-10 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 1093	Acknowledged

Description

The return values of function `addLiquidityETH` are not properly handled.

```
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
);
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-11 | Centralized Risk In `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/HydroLink/contracts/HydroLink.sol (1230346): 1098	Partially Resolved

Description

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `HydroLink-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens.If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. 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, f.e. Multisignature wallets.

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

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

`[HydroLink]`: Upon token deployment, the HydroLink team immediately identified _liquidity to be a material security risk for token holders. Consistent with the SafeMoon protocol, the LP tokens were being received by the _owner's address. In response to the growing LP balance of _owner address and the team’s recognition of this problem, the LP tokens were transferred to the contract address (zero address). Further, the additional LP tokens (0.75 LP) that were received by _owner address were locked (https://dxsale.app/app/v3/dxlockview?id=0&add=0xFE4A26e62b18EC1c1a9F1739BFf04c58F85fb99b&type=lplock&chain=BSC)

HLH-12 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 1112	Acknowledged

Description

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in `else` .

Recommendation

The following code can be removed:

```
... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferStandard(sender, recipient, amount);
} ...
```

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-13 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	projects/HydroLink/contracts/HydroLink.sol (1230346): 882, 886	Acknowledged

Description

The variables `_taxFee` and `_liquidityFee` should not exceed 100 respectively, and the sum of them should not exceed 100.

Recommendation

We advise the client to check that the variables `_taxFee` and `_liquidityFee` like as follows:

```
require(taxFee <= 100, "TaxFee exceed 100");
```

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-14 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	projects/HydroLink/contracts/HydroLink.sol (1230346): 422, 431, 437, 442, 450, 756, 760, 764, 768, 777, 782, 786, 791, 797, 802, 807, 811, 815, 824, 841, 874, 878, 890, 894, 898, 958, 989	Acknowledged

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-15 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/HydroLink/contracts/HydroLink.sol (1230346): 422, 431, 442, 841, 851, 874, 878, 882, 886, 890, 894, 898, 958	Acknowledged

Description

To bridge the gap in trust between the administrators need to express a sincere attitude regarding the considerations of the administrator team's anonymity.

The `owner` has the responsibility to notify users about the following capabilities:

- transfer ownership of the contract through `transferOwnership()`
- renounce ownership of the contract through `renounceOwnership()`
- lock the contract through `lock()`
- exclude from reward through `excludeFromReward()`
- include from reward through `includeInReward()`
- exclude from fee through `excludeFromFee()`
- include from fee through `includeInFee()`
- set tax fee percent through `setTaxFeePercent()`
- set liquidity fee percent through `setLiquidityFeePercent()`
- set `numTokensSellToAddToLiquidity` through `setNumTokensSellToAddToLiquidity()`
- set `_maxTxAmount` through `setMaxTxPercent()`
- enable `swapAndLiquifyEnabled` through `setSwapAndLiquifyEnabled()`
- transfer BNB in the contract to itself through `claimTokens()`

Recommendation

We advise the client to carefully manage the privileged account's private keys 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 suggestions that would also mitigate the potential risks at the different levels in terms of the 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

`[HydroLink]`: HydroLink token is a fair launched token that is governed by a central board. The management team firmly understands the roles and responsibilities for which we are accountable.

HLH-16 | Not Update `rOwned`

Category	Severity	Location	Status
Logical Issue	Medium	projects/HydroLink/contracts/HydroLink.sol (1230346): 851	Acknowledged

Description

When an account is included in the reward list through `includeInReward()`, the account's `rOwned` balance is not updated to reflect the change in `rTotal`. This may lead to a miscalculation of the account's deserved reward. For example, when an account on the reward list is excluded, its `tOwned` balance would be locked for a period of time before the account is included back again. However, in that situation, the universal `rate` is likely to decrease in that period so that by reflecting the `rOwned` the account would receive more tokens than deserved which in effect cancels out that excluded period.

Recommendation

We advise the client to keep `tOwned` unchanged and update `rOwned` accordingly in `includeInReward()`

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

HLH-17 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Medium	projects/HydroLink/contracts/HydroLink.sol (1230346): 1079, 1093	Acknowledged

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens()`
- `uniswapV2Router.addLiquidityETH()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

`[HydroLink]`: Given the deployed contract cannot be updated, decided to retain the code base unchanged.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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

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.