CERTIK

Security Assessment

Defrost Finance II

Dec 16th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- GLOBAL-01 : Unlocked Compiler Version
- GLOBAL-02 : Third Party Dependencies
- GLOBAL-03 : Function Visibility Optimization
- GLOBAL-04 : Centralization Risk
- GLOBAL-05 : Hardcode Address
- GLOBAL-06 : Unused Return Values
- GLOBAL-07 : Set `constant` to Variables
- GLOBAL-08 : Potential Reentrancy Attack
- GLOBAL-09 : Potential Sandwich Attacks
- GLOBAL-10 : Inconsistent Logic For function `_setReward`
- GLOBAL-11 : Discussion For Function `compound()`
- GLOBAL-12 : Potential Flashloan Attack
- TVV-01 : Lack of Input Validation
- TVV-02 : Incompatibility With Deflationary Tokens
- TVV-03 : Discussion For `slipeRate`
- TVV-04 : Lack of keyword

Appendix

Disclaimer

About

Summary

This report has been prepared for Defrost Finance II to discover issues and vulnerabilities in the source code of the Defrost Finance II 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	Defrost Finance II
Platform	Avalanche C-Chain
Language	Solidity
Codebase	https://github.com/DefrostFinance/defrost-finance-contract/tree/master/contracts/superVault
Commit	4d9bcc6bb4d8f2ca09a048ea51e57e22f7523aaf

Audit Summary

Delivery Date	Dec 16, 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	1	0	0	1	0	0
● Medium	2	0	0	1	0	1
● Minor	6	0	0	4	0	2
● Informational	7	0	0	4	0	3
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ERC	superVault/ERC20.sol	17065068ada68777512f7c9b8e9c191b78c31e303c27d35b92237d182ee04e37
IBC	superVault/IBenqiCompound.sol	b6118f73fe74553c7c5136d9149d47bf2088cfc91590c361c29e0c7ca8ef2f93
ICG	superVault/ICurveGauge.sol	29ba7e100f29f788fbfff9c7f96db8207558d5098b341ca3bdd9b3c663928ac8
CAV	superVault/superCurveAv3.sol	fed09c46ae7f91b34db7b8c3f7f7bfa77952bd5be73eff859bb0a7e5a8f3291f
QAV	superVault/superQiAvax.sol	612c4f0924b0a81c32d9e43dcfd811a6dd7a8c779624a044ec3f0a39bbfd9b12
QEV	superVault/superQiErc20.sol	6f08f9f26c64bcc1c076d9be94c187921b602cda15174aeef57f27ccf57a3c78
QTV	superVault/superQiToken.sol	ddefb91039bc447f5e867208073c7e075e3750afe85b8b47ac52f73ab9ac0310
TVV	superVault/superToken.sol	cb0ba13e92509e069da93ed6b4f62e0fbd5e564ca1535dfe54b2c993ea410db6

It should be noted that the system design includes a number of economic arguments and assumptions. These were explored to the extent that they clarified the intention of the code base, but we did not audit the mechanism design itself. Note that financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol. The correctness of the financial model is not in the scope of the audit.

Note that this audit only includes the contracts in the stated scope while the files outside the scope are treated as black boxes and are assumed to be functionally correct.

To bridge the trust gap between owner and users, the owner needs to express a sincere attitude with the consideration of the administrator team's anonymousness.

The `origin` of `superToken` has the responsibility to notify users about the following capabilities:

- set `FeePool` through `setFeePoolAddress()`
- set `slipRate` through `setSlipRate()`
- set `feeRate` through `setFeeRate()`
- set `swapRoutingPath` through `setSwapRoutingPathInfo()`

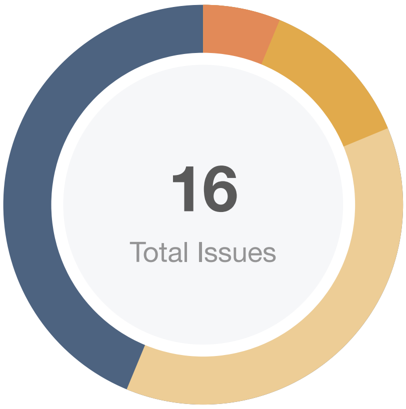
The `origin` of `superQiToken` has the responsibility to notify users about the following capabilities:

- set `rewardInfos` through `setReward()`

The `origin` of `superCurveAv3` has the responsibility to notify users about the following capabilities:

- set `rewardInfos` through `setReward()`

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**1** (6.25%)
🟨 **Medium**	**2** (12.50%)
🟨 **Minor**	**6** (37.50%)
🟦 **Informational**	**7** (43.75%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version	Language Specific	● Informational	ⓘ Acknowledged
GLOBAL-02	Third Party Dependencies	Volatile Code	● Minor	ⓘ Acknowledged
GLOBAL-03	Function Visibility Optimization	Gas Optimization	● Informational	✓ Resolved
GLOBAL-04	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
GLOBAL-05	Hardcode Address	Logical Issue	● Informational	ⓘ Acknowledged
GLOBAL-06	Unused Return Values	Gas Optimization	● Informational	ⓘ Acknowledged
GLOBAL-07	Set `constant` to Variables	Gas Optimization	● Informational	✓ Resolved
GLOBAL-08	Potential Reentrancy Attack	Logical Issue	● Medium	ⓘ Acknowledged
GLOBAL-09	Potential Sandwich Attacks	Logical Issue	● Minor	✓ Resolved
GLOBAL-10	Inconsistent Logic For function `_setReward`	Volatile Code	● Minor	ⓘ Acknowledged
GLOBAL-11	Discussion For Function `compound()`	Logical Issue	● Informational	ⓘ Acknowledged
GLOBAL-12	Potential Flashloan Attack	Logical Issue	● Medium	✓ Resolved
TVV-01	Lack of Input Validation	Volatile Code	● Minor	ⓘ Acknowledged
TVV-02	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	ⓘ Acknowledged

ID	Title	Category	Severity	Status
TVV-03	Discussion For `slipeRate`	Inconsistency	● Informational	✓ Resolved
TVV-04	Lack of keyword	Logical Issue	● Minor	✓ Resolved

GLOBAL-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	● Informational	Global	ⓘ Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract 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 an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify 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.

GLOBAL-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	Global	ⓘ 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.

- `superQiErc20`
- `superQiAvax`
- `superCurveAv3`
- `ChainLinkOracle`

Recommendation

We encourage the team to constantly monitor the status of those 3rd parties to mitigate negative outcomes when unexpected activities are observed.

Alleviation

`[Client]`: Third Part is limited to trusted like Curve Finance, Trader Joe on Avalanche, ChainlinkOracle.

GLOBAL-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	● Informational	Global	✓ Resolved

Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope:

- `ERC20.name()`
- `ERC20.symbol()`
- `ERC20.decimals()`
- `ERC20.balanceOf()`
- `ERC20.transfer()`
- `ERC20.allowance()`
- `ERC20.approve()`
- `ERC20.transferFrom()`
- `ERC20.increaseAllowance()`
- `ERC20.decreaseAllowance()`
- `superToken.enter()`
- `superToken.leave()`
- `superQiAvax.compound()`
- `superQiErc20.compound()`
- `superCurveAv3.compound()`

The functions that are never called internally within the contract should have external visibility.

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

The client heeded our advice and resolved this issue in commit : 0c741bae373e01425931922ba814c53341d57ec2.

GLOBAL-04 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	Global	ⓘ 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 `origin` of `superToken` has the responsibility to notify users about the following capabilities:

- set `FeePool` through `setFeePoolAddress()`
- set `slipRate` through `setSlipRate()`
- set `feeRate` through `setFeeRate()`
- set `swapRoutingPath` through `setSwapRoutingPathInfo()`

The `origin` of `superQiToken` has the responsibility to notify users about the following capabilities:

- set `rewardInfos` through `setReward()`

The `origin` of `superCurveAv3` has the responsibility to notify users about the following capabilities:

- set `rewardInfos` through `setReward()`

Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g. Multisignature wallets.

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

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

Alleviation

`[Client]`: Like Defrost vault, modifier `onlyOrigin()` have a multi-signature check

```
    modifier onlyOrigin() {
        require (isOrigin(),"proxyOwner: caller is not the tx origin!");
        checkMultiSignature();
        _;
}
```

GLOBAL-05 | Hardcode Address

Category	Severity	Location	Status
Logical Issue	● Informational	Global	ⓘ Acknowledged

Description

There are many hardcode addresses in this codebase.

Recommendation

We advise changing to the correct address before the contract is deployed onto blockchain.

Alleviation

`[Client]`: We have checked all of the addresses. They are correct.

GLOBAL-06 | Unused Return Values

Category	Severity	Location	Status
Gas Optimization	● Informational	Global	ⓘ Acknowledged

Description

The return value `path` is not used.

- `superQiToken.getSwapRouterPath`
- `superCurveAv3.getSwapRouterPath`

Recommendation

We advise the client to remove them.

Alleviation

`[Client]`: The functions have been used in trader joe swap function.

```
function swapTraderJoe(address token,uint256 sellLimit)internal{
    if(token == underlying){
        return;
    }
    uint256 balance = IERC20(token).balanceOf(address(this));
    if (balance < sellLimit){
        return;
    }
    address[] memory path = getSwapRouterPath(token);
    uint[] memory amountOut = IJoeRouter01(traderJoe).getAmountsOut(balance, path);
    uint256 minOut = amountOut[amountOut.length-1]*slipRate/10000;

IJoeRouter01(traderJoe).swapExactTokensForAVAX(balance,minOut,path,address(this),block.ti
mestamp+30);
}
```

GLOBAL-07 | Set `constant` to Variables

Category	Severity	Location	Status
Gas Optimization	● Informational	Global	✓ Resolved

Description

The following variables could be declared as `constant` since these state variables are never modified.

- `superToken.WAVAX` #25
- `superQiToken.compounder` #24
- `superQiToken.traderJoe` #25
- `superCurveAv3.underlying` #12
- `superCurveAv3.av3Crv` #22
- `superCurveAv3.traderJoe` #23

Recommendation

We recommend to declare these variables as `constant`.

Alleviation

The client heeded our advice and resolved this issue in commit : 0c741bae373e01425931922ba814c53341d57ec2.

GLOBAL-08 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	● Medium	Global	ⓘ Acknowledged

Description

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

- `superToken.enter()`
- `superToken.leave()`
- `superQiErc20.compound()`
- `superQiAvax.compound()`
- `superCurveAv3.compound()`

Recommendation

We advise the client to apply OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

`[Client]`: Function `enter()` and `leave()` have been reentrancy protected. Function Compound() is free. commit : 0c741bae373e01425931922ba814c53341d57ec2.

GLOBAL-09 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	● Minor	Global	✓ Resolved

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens 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:

- `IJoeRouter01.swapExactTokensForAVAX()`
- `IJoeRouter01.swapExactAVAXForTokens()`
- `IJoeRouter01.swapExactTokensForTokens()`

Recommendation

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

Alleviation

The client revised the code and resolved this issue in commit : bbbc3fd452d9a13370f336ab8b3612ab33379994.

GLOBAL-10 | Inconsistent Logic For function `_setReward`

Category	Severity	Location	Status
Volatile Code	● Minor	Global	ⓘ Acknowledged

Description

The `_setReward` function is to add reward tokens for the user, If the variable `index` is less than the length of the array `rewardInfos`, the reward tokens will be updated without verifying whether the tokens are the same as before. If it is not the same, the previous token authorization needs to be canceled, and the amount authorized to `traderJoe` is too large.

- `superQiToken._setReward()`
- `superCurveAv3._setReward()`

Recommendation

We advise the client to recheck the function.

Alleviation

`[Client]`: We want to change `rewardInfo` if the reward token has been changed in the mint pool.

GLOBAL-11 | Discussion For Function `compound()`

Category	Severity	Location	Status
Logical Issue	● Informational	Global	ⓘ Acknowledged

Description

The `compound` function is to extract the reward of `rewardInfos`, exchange the `underlying` token through third-party DEX, charge an amount of fee, and then invoke the `mint` method of `stakeToken`. We would like to enquire the purpose of the functionality of this `mint` function?

- `superQiErc20.compound()`
- `superQiAvax.compound()`
- `superCurveAv3.compound()`

Alleviation

`[Client]`: Mint is QiToken's function. It will be staked and obtained QiToken.

GLOBAL-12 | Potential Flashloan Attack

Category	Severity	Location	Status
Logical Issue	● Medium	Global	✓ Resolved

Description

Flash loans are a way to borrow large amounts of money for a certain fee. The requirement is that the loans need to be returned within the same transaction in a block. If not, the transaction will be reverted.

An attacker can use the borrowed money as the initial funds for an exploit to enlarge the profit and/or manipulate the token price in the decentralized exchanges.

We find that the following functions rely on price calculations that are based on-chain, meaning that they would be susceptible to flash-loan attacks by manipulating the price of given pairs to the attacker's benefit.

- `superQiErc20.swapTraderJoe()`
- `superQiAvax.swapTraderJoe()`
- `superCurveAv3.swapTraderJoe()`

Recommendation

If a project requires price references, it needs to be cautious of flash loans that might manipulate token prices. To minimize the chance of happening, we recommend the client to consider following according to the project's business model:

1. Use multiple reliable on-chain price oracle sources, such as Chainlink and Uniswap.
2. If the business model allows, restrict the function caller to be a non-contract/EOA address.
3. Flash loans only allow users to borrow money within a single transaction. If the contract use cases are allowed, force critical transactions to span at least two blocks.

Alleviation

The client used Chainlink oracles to obtain asset prices and and resolved this issue in commit : bbbc3fd452d9a13370f336ab8b3612ab33379994.

TVV-01 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	● Minor	projects/superVault/contracts/superVault/superToken.sol (243bf3c): 35, 34	ⓘ Acknowledged

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
    require(FeePool != address(0), "FeePool is 0");
...
```

Alleviation

No alleviation

TVV-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	projects/superVault/contracts/superVault/superToken.sol (243bf3c): 61	ⓘ Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user stakes 100 deflationary tokens (with a 10% transaction fee) in a MasterChef, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

The MasterChef takes the pool token balance(the `lpSupply`) into account when calculating the users' reward. An attacker can repeat the process of deposit and withdraw to lower the token balance(`lpSupply`) in a deflationary token pool and cause the contract to increase the reward amount.

Reference: https://thoreum-finance.medium.com/what-exploit-happened-today-for-gocerberus-and-garuda-also-for-lokum-ybear-piggy-caramelswap-3943ee23a39f

Recommendation

We advise the client to regulate 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

`[Client]`: The set of tokens are not supported.

TVV-03 | Discussion For `slipeRate`

Category	Severity	Location	Status
Inconsistency	● Informational	projects/superVault/contracts/superVault/superToken.sol (243bf3c): 21	✓ Resolved

Description

The value of `slipRate` should be a number smaller than 5000 in the `setSlipRate()` function. But the initial value is 9500.

Recommendation

We advise the client to recheck the value.

Alleviation

The client revised the code and resolved this issue in commit : 0c741bae373e01425931922ba814c53341d57ec2.

TVV-04 | Lack of keyword

Category	Severity	Location	Status
Logical Issue	● Minor	projects/superVault/contracts/superVault/superToken.sol (243bf3c): 99	✓ Resolved

Description

`SetSwapRoutingPath` is missing the keyword `emit`.

Recommendation

We advise the client to add the keyword `emit`.

Alleviation

The client heeded our advice and resolved this issue in commit : 0c741bae373e01425931922ba814c53341d57ec2.

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.

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.

Inconsistency

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

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK