# Code Security Assessment

# Solv Protocol 3

Jan 21st, 2022

# Table of Contents

# Summary

This report has been prepared for Solv Protocol 3 to discover issues and vulnerabilities in the source code of the Solv Protocol 3 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.

Additionally, this audit is based on a premise that all external contracts were implemented safely.

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** | Solv Protocol 3 |
| **Platform** | ethereum, bsc, polygon |
| **Language** | Solidity |
| **Codebase** | https://github.com/solv-finance/solv-v2-ivo |
| **Commit** | bf6dac361e4b6fb4671ccc68c8f3bd1d155da545<br>b207d5ef3077fd63fb7d78e769a56a800903d303 |

## Audit Summary

| | |
|---|---|
| **Delivery Date** | Jan 21, 2022 |
| **Audit Methodology** | Static Analysis, Manual Review |

## Vulnerability Summary

| Vulnerability Level | Total | Pending | Declined | Acknowledged | Partially Resolved | Resolved |
|---|---|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 | 0 | 0 |
| Major | 2 | 0 | 0 | 1 | 0 | 1 |
| Medium | 1 | 0 | 0 | 0 | 0 | 1 |
| Minor | 3 | 0 | 0 | 1 | 0 | 2 |
| Informational | 9 | 0 | 0 | 1 | 0 | 8 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

# Audit Scope

| ID | File | SHA256 Checksum |
| --- | --- | --- |
| SCM | solv-3/markets/convertible-marketplace/contracts/SolvConvertibleMarket.sol | 6a292d7370a46811bb015a42ff8b346c8741639e2e08afe9d61f108db863060e |
| ICO | solv-3/markets/convertible-offering-market/contracts/InitialConvertibleOfferingMarket.sol | 1a6f863a8b2a32f713bcc6259fc690a5bc7172edd9f26a15fe3bcaf01a4fee3e |
| CPO | solv-3/vouchers/convertible-voucher/contracts/oracle/ChainlinkPriceOracle.sol | f22e65fed853a51f112d97b177b6dae6c5c84770fbefe840115a96281dcbed33 |
| MPO | solv-3/vouchers/convertible-voucher/contracts/oracle/ManualPriceOracle.sol | 6cf35afafb435ea3d71933d16224ddee5571b106228e0f28b42738de370afc80 |
| POM | solv-3/vouchers/convertible-voucher/contracts/oracle/PriceOracleManager.sol | 12deabfdad48118bc049f7ca724837ab5eb21a0860a47e7c1f281a36eda9c604 |
| CPC | solv-3/vouchers/convertible-voucher/contracts/ConvertiblePool.sol | 0861f48bb248795dad5fc59355f32a7cbbad408cd142face010381a9e328ea7c |
| CVC | solv-3/vouchers/convertible-voucher/contracts/ConvertibleVoucher.sol | a032d15ceb289986901625fbade23fee52bf8ccbae44ef0b2c28bd5a19020307 |

# Privileged Functions

The contract contains the following privileged functions that are restricted by some modifiers. They are used to modify the contract configurations and address attributes. We grouped these functions below:

## The `onlyAdmin` modifier:

Contract `SolvConvertibleMarketplace`:

- _addMarket( address voucher_, uint128 precision_, uint8 feePayType_, uint8 feeType_, uint128 feeAmount_, uint16 feeRate_ )
- _removeMarket(address voucher_)
- _setCurrency(address currency_, bool enable_)
- _withdrawFee(address currency_, uint256 reduceAmount_)
- setAllowAddressManager( address voucher_, address[] calldata managers_, bool resetExisting_ )
- _setSolver(ISolver newSolver_)

Contract `ConvertiblePool`:

- setFundCurrency(address fundCurrency_, bool enable_)
- setVoucher(address newVoucher_)

Contract `ConvertibleVoucher`:

- setVoucherDescriptor(address newDescriptor)
- setSolver(ISolver newSolver_)

Contract `ChainlinkPriceOracle`:

- setJobId(bytes32 jobId_)
- setOraclePayment(uint256 payment_)
- setTokenId(address underlying_, uint256 tokenId_)
- setPriceOracleManager(address manager_)
- setPendingAdmin(address newPendingAdmin)

Contract `ManualPriceOracle`:

- _setPrice( address underlying_, uint64 maturity_, int256 price_ )

Contract `PriceOracleManager`:

- _setVoucherOracle(address voucher_, IPriceOracle oracle_)

- _setDefaultOracle(IPriceOracle newOracle_)
- _setDefaultPricerPeriod(uint64 pricePeriod_)
- _setPricePeriod(address voucher_, uint64 pricePeriod_)

## The `onlyAllowAddressManager` modifier:

Contract `SolvConvertibleMarketplace`:

- _addAllowAddress( address voucher_, address[] calldata addresses_, bool resetExisting_ )
- _removeAllowAddress( address voucher_, address[] calldata addresses_ )

## The `onlyVoucher` modifier:

Contract `ConvertiblePool`:

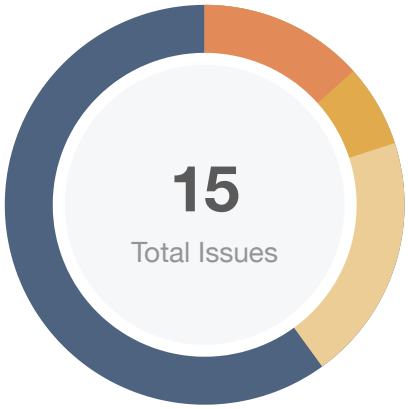
- createSlot( address issuer_, address fundCurrency_, uint128 lowestPrice_, uint128 highestPrice_, uint64 effectiveTime_, uint64 maturity_, uint8 collateralType_ )
- mintWithUnderlyingToken( address minter_, uint256 slot_, uint256 tokenInAmount_ )
- claim( uint256 slot_, address to_, uint256 claimValue_ )

## The `onlyPriceOracleManager` modifier:

Contract `ChainlinkPriceOracle`:

- refreshPrice( address underlying_, uint64 fromDate_, uint64 toDate_ )

# Findings




| | |
|---|---|
| **Critical** | **0** (0.00%) |
| **Major** | **2** (13.33%) |
| **Medium** | **1** (6.67%) |
| **Minor** | **3** (20.00%) |
| **Informational** | **9** (60.00%) |
| **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **GLOBAL-01** | Centralization Risk | **Centralization / Privilege** | **Major** | Acknowledged |
| GLOBAL-02 | Function Visibility Optimization | Gas Optimization | Informational | Resolved |
| GLOBAL-03 | Missing Emit Events | Coding Style | Informational | Resolved |
| GLOBAL-04 | Finance Model | Logical Issue | Minor | Acknowledged |
| GLOBAL-05 | Discussion For Contract `ManualPriceOracle` | Logical Issue | Informational | Acknowledged |
| CPC-01 | `totalValue` Should Be Accumulated In `mintWithUnderlyingToken()` | Logical Issue | Major | Resolved |
| CPC-02 | `withdrawCurrencyAmount` And `withdrawTokenAmount` Should Be Deducted In `withdraw()` | Logical Issue | Minor | Resolved |
| CPC-03 | Missing Input Validation | Logical Issue | Informational | Resolved |
| CPO-01 | Missing Input Validation | Logical Issue | Informational | Resolved |
| CVC-01 | Potential Reentrancy Attack | Logical Issue | Informational | Resolved |
| CVC-02 | Missing Input Validation | Logical Issue | Informational | Resolved |
| ICO-01 | Unused Enum | Logical Issue | Informational | Resolved |

| ID | Title | Category | Severity | Status |
| --- | --- | --- | --- | --- |
| SCM-01 | Duplicated Fee Charges | Logical Issue | Medium | Resolved |
| SCM-02 | Missing Input Validation | Logical Issue | Informational | Resolved |
| SCM-03 | Missing Input Validation | Logical Issue | Minor | Resolved |

## GLOBAL-01 | Centralization Risk

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Centralization / Privilege | Major | Global | Acknowledged |

## Description

In the contract `SolvConvertibleMarketplace`, the role `admin` has the authority over the following function:

- _addMarket()
- _removeMarket()
- _setCurrency()
- _withdrawFee()
- setAllowAddressManager()
- _setSolver()

In the contract `SolvConvertibleMarketplace`, the role `allowAddressManager` has the authority over the following function:

- _addAllowAddress()
- _removeAllowAddress()

In the contract `ChainlinkPriceOracle`, the role `admin` has the authority over the following function:

- fulfill()
- setJobId()
- setOraclePayment()
- setTokenId()
- setPriceOracleManager()
- setPendingAdmin()

In the contract `ManualPriceOracle`, the role `admin` has the authority over the following function:

- _setPrice()
- setPendingAdmin()

In the contract `PriceOracleManager`, the role `admin` has the authority over the following function:

- _setVoucherOracle()
- _setDefaultOracle()

- _setDefaultPricerPeriod()
- _setPricePeriod()

In the contract `ConvertibleVoucher`, the role `admin` has the authority over the following function:

- setVoucherDescriptor()
- setSolver()

Any compromise to these accounts may allow the hacker to manipulate the project through these functions.

## 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.

## Alleviation

The client response:
Regarding centralization and authority risk, there are two parts of the contract that use authority: one is the need for business management, and some businesses need to dynamically adjust parameters; the other is to use the management mechanism of the upgradeable contract framework. This issue will not be revised for the time being. Later, depending on the situation, the management rights will be transferred to the timelock contract or voting mechanism, and finally delegated to the community.

## GLOBAL-02 | 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. The functions that are never called internally within the contract should have external visibility.

**contract `SolvConvertibleMarketplace`**

- `_addMarket()` in L672
- `_removeMarket()` in L699
- `_setCurrency()` in L705
- `_withdrawFee()` in L710

**contract `ManualPriceOracle`**

- `_setPrice()` in L27

## 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 team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## GLOBAL-03 | Missing Emit Events

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | Global | Resolved |

## Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

**contract `SolvConvertibleMarketplace`**

- `_addAllowAddress()`
- `_removeAllowAddress()`
- `setAllowAddressManager()`

**contract `ChainlinkPriceOracle`**

- `refreshPrice()`
- `setJobId()`
- `setOraclePayment()`
- `setTokenId()`
- `setPriceOracleManager()`

**contract `ManualPriceOracle`**

- `_setPrice()`

**contract `PriceOracleManager`**

- `_setDefaultPricerPeriod()`
- `_setPricePeriod()`

## Recommendation

We advise the client to add events for sensitive actions, and emit them in the function.

## Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## GLOBAL-04 | Finance Model

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | Global | Acknowledged |

### Description

This audit includes three parts of the contract: oracle, voucher and market.

### Oracle

Oracle contracts include the ChainlinkPriceOracle, ManualPriceOracle and PriceOracleManager contracts. The admin of the PriceOracleManager contract can set an oracle for each voucher, the price of the ChainlinkPriceOracle contract is obtained from ChainlinkOracle, and the price of the ManualPriceOracle contract is set by the contract admin.

### Voucher

ConvertibleVoucher is the purpose of the project party for financing through mortgage Token. The mortgage amount is within a certain range, there is no liquidation, and only settlement is due according to certain rules.
The issuer of this product is the client, and the client also sets the price range and lock-in period to mint Convertible Vouchers of a certain value (the maximum number of tokens mortgaged), and then sell them to users at a certain discount through `InitialConvertibleOfferingMarket`. After the user's purchase is settled, the user can obtain the underlying Token or stable token equal to the voucher value at the settlement price (depending on whether the issuer is refunded), and the settlement price determines whether the user loses or makes a profit.

1. If the settlement price is within the price range of the issue, the user will obtain equivalent underlying token or fund currency;
2. If the settlement price is above the price range, the user will make a profit;
3. If the settlement price is below the price range, the user will be lost.

### Market

The holder of the voucher can choose to sell the voucher in the market at a stable price or a decaying price. The admin of the contract decides the type of charge (buyer pays or seller pays) and the fee of charge (fixed fee or percentage of transaction amount). It should be noted that there is no limit to the percentage charge (0～100%).

## Recommendation

We recommend the client to publish the financial models to the community.

## Alleviation

No alleviation.

## GLOBAL-05 | Discussion For Contract `ManualPriceOracle`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | Global | Acknowledged |

## Description

According to the logic, some contracts will obtain the price of the token from the price oracle, so the price oracle contract affects the user's income, but the price in the `ManualPriceOracle` contract is determined by the admin of the contract. Could you please answer the question:

1. What is the role of the admin of the contract? the UA or the governance organization or others?
2. Why does the project needs `ManualPriceOracle` contract?

## Recommendation

We recommend the client do not use the contact.

## Alleviation

The client response:
ManualPriceOracle is just a test contract. The contracts currently online will use Chainlink Oracle or UniSwap Oracle.

## CPC-01 | `totalValue` Should Be Accumulated In `mintWithUnderlyingToken()`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | solv-3/vouchers/convertible-voucher/contracts/ConvertiblePool.sol: 181 | Resolved |

### Description

The user can pledge `underlyingToken` to obtain NFT through `ConvertibleVoucher.mint()`. The total value of NFT is equal to the amount of pledged tokens * lowest price. The function generates the `slot` according to the set parameters. The total value of NFT under the same slot should be accumulated, not overwritten and updated.

### Recommendation

We recommend the client modify as below:
mintWithUnderlyingToken():

```
181    slotDetail.totalValue = slotDetail.totalValue.add(totalValue);
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## CPC-02 | `withdrawCurrencyAmount` And `withdrawTokenAmount` Should Be Deducted In `withdraw()`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | solv-3/vouchers/convertible-voucher/contracts/ConvertiblePool.sol: 286 | Resolved |

### Description

When the issuer withdraws assets from the pool, `withdrawTokenAmount` and `withdrawCurrencyAmount` should be deducted from `slotBalances[slot_][underlyingToken]` and `slotBalances[slot_][slotDetail.fundCurrency]`, the balance of slot is not updated in `withdraw()`.

### Recommendation

We recommend the client modify as below:

withdraw():

```
286    uint256 reservedCurrencyAmount = slotBalances[slot_][slotDetail.fundCurrency];
287    uint256 reservedTokenAmount = slotBalances[slot_][underlyingToken];
288    slotBalances[slot_][underlyingToken] =
289                reservedTokenAmount.sub(withdrawTokenAmount);
290    slotBalances[slot_][slotDetail.fundCurrency] =
291                reservedCurrencyAmount.sub(withdrawCurrencyAmount);
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## CPC-03 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/vouchers/convertible-voucher/contracts/ConvertiblePool.sol: 119 | Resolved |

### Description

The given input is missing the sanity check.

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below: validateSlotParams():

```
119  require(effectiveTime_ > 0), "effectiveTime_ should be grater than 0.");
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## CPO-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/vouchers/convertible-voucher/contracts/oracle/ChainlinkPriceOracle.sol: 157 | Resolved |

### Description

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

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below: setPriceOracleManager():

```
157  require(manager_ != address(0), "manager_ can not be 0 address.");
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## CVC-01 | Potential Reentrancy Attack

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/vouchers/convertible-voucher/contracts/ConvertibleVoucher.sol: 47 | Resolved |

### 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.

### Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## CVC-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/vouchers/convertible-voucher/contracts/ConvertibleVoucher.sol: 190 | Resolved |

### Description

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

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:
setVoucherDescriptor():

```
190  require(newDescriptor != address(0), "newDescriptor can not be 0 address.");
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## ICO-01 | Unused Enum

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/markets/convertible-offering-market/contracts/InitialConvertibleOfferingMarket.sol: 40~44 | Resolved |

### Description

The enum `TimeType` is declared but never used in the contract.

### Recommendation

We recommend to remove the unused enum.

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## SCM-01 | Duplicated Fee Charges

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Medium | solv-3/markets/convertible-marketplace/contracts/SolvConvertibleMarket.sol: 375~380, 430~435, 501~504 | Resolved |

### Description

In the functions `buyByAmount()` and `buyByUnits()`, when `FeePayTyper` is `BUYER_PAY`, `unints_` is calculated after deducting fees from `amount_`. But in the `_buy()` function, the number of tokens actually transferred is `amount_ + fee`, that is, the fee is actually deducted 2 times.

### Recommendation

We recommend the client to redesign this part of logic.

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## SCM-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | solv-3/markets/convertible-marketplace/contracts/SolvConvertibleMarket.sol: 672 | Resolved |

### Description

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

### Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:
_addMarket():

```
672  require(voucher_ != address(0), "voucher_ can not be 0 address.");
```

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

## SCM-03 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | solv-3/markets/convertible-marketplace/contracts/SolvConvertibleMarket.sol: 685 | Resolved |

### Description

`feeRate_` lacks the restriction, it is better to add the upper limit for `feeRate_`.

### Recommendation

We recommend the client add the restriction for `feeRate_` to avoid error operations.

### Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit b207d5ef3077fd63fb7d78e769a56a800903d303.

# 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.

### 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.