# Code Security Assessment

# Accessifi

Jan 10th, 2022

# Table of Contents

# Summary

This report has been prepared for Accessifi to discover issues and vulnerabilities in the source code of the Accessifi 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 | Accessifi |
| Platform | bsc |
| Language | Solidity |
| Codebase | https://github.com/Accessifi/Accessifi-core |
| Commit | e3a8a6828fdff9f18621b3069a6d2a81e4c46ec5<br>29e83d9fdbf4660c69176921e7247a2e41c9275e |

## Audit Summary

| | |
|---|---|
| Delivery Date | Jan 10, 2022 |
| 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 | 2 | 0 | 0 | 1 | 0 | 1 |
| Medium | 0 | 0 | 0 | 0 | 0 | 0 |
| Minor | 2 | 0 | 0 | 0 | 0 | 2 |
| Informational | 14 | 0 | 0 | 3 | 0 | 11 |
| Discussion | 0 | 0 | 0 | 0 | 0 | 0 |

## Audit Scope

| ID | File | SHA256 Checksum |
|---|---|---|
| IAU | interfaces/iauction.sol | 1cfbf35cb1bfa33685273835dcaac07249c85f8c1387200692c666ade1f93bef |
| IFA | interfaces/ifactory.sol | 7d566baa43ab1712e2c5411a16132ce9ae97f13148d1e5517c62700363514e7b |
| IMA | interfaces/imarket.sol | 7fce9a5df838cd600363ede5e178854d1637355fb3fd96aaf6efa5d3029b6148 |
| ISA | interfaces/isaft.sol | de4e9561b0960b9f54fef19f31af8e8ce92003db1733792acacd7a7113914cd1 |
| IVE | interfaces/ivesting.sol | 21ec0d059e8dc2f3dc850f4d37e1264dc9047e02f00ca85f5eae10de31de5530 |
| IWE | interfaces/iweth.sol | af35a7c2df996b0d200e0e3ddf7f257f9f4816034c54ee90d3219f18fe07cd46 |
| BAS | markets/basemarket.sol | a5f8d30c6ac28054c1f7a081432f8fb99f4968af539ec595de33acf32bb1d829 |
| DUT | markets/dutchauction.sol | b90d6159886831cef354a558200a8476d82e0df49466b3856b5ce0dab701019d |
| ENG | markets/englishauction.sol | 51526c71cc087ca5f9db544fc87885622d00b076eb78d1e01b2dcd0c92331660 |
| FIX | markets/fixedprice.sol | 0746665906a595488f9434d4039fd5094ff7bd4f5b6c693863bd6c726c391559 |
| ACP | vestings/basevesting.sol | 3896c84a301a694b444ab3fee5d4dd886d2f9db053b69676f6120f2293f2ca1c |
| AKP | vestings/linearly.sol | 66ffeb898ea4dd7374922d83a7ba39df791a96923d7b4b68cbf4d645549d99b2 |
| CKP | vestings/onetime.sol | 009db5d28276c66ff443a72ee41946248a91b5d7dc48540fe74b03784ff6c3d3 |
| STA | vestings/staged.sol | 9d92f07c730da4b46357e70cde74399e72fad02db03ee1ae9c343269d46ffce2 |
| NFT | nftfund.sol | c2429df4cf01f049c87a6b3645baacb0117be7ae836b2457e31843621e29ef2d |
| SAF | saft.sol | d6ff393635f2c2747bd2fc38c3e3353c44d8a9031e985c3db7edf61348c6bf61 |
| SAT | saftfactory.sol | 239b37024ae6b9addb03c4baad329356587dc670345c57d3c2fa794d23602ee0 |

# Understandings

## Overview

`Accessifi` is the SAFT NFT Marketplace. Users can create SAFT. If they create SAFT with tokens, and then the owner of the NFT can claim tokens based on the following strategies: 1. Linear release. 2. One-time release. 3. Phased release.

There are three types of NFT trading markets available in `Accessifi`:

- 1.Dutch Auction
  Over time, the NFT auction price will gradually decrease.
- 2.EnglishAuction
  Multiple users participate in the NFT auction, and the NFT owner accepts the bid.
- 3.FixedPrice
  Users purchase or sell NFTs through pending orders.

## 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 `onlyDev` modifier:

Contract `BaseMarket`:

- setDevAddr(address _devAddr)
- setFeeRatio(uint256 _feeRatio)

Contract `BaseMarket`:

- setPriceIncrRatio(uint256 _priceIncrRatio)

Contract `Saft`:

- transferDevAddr(address _newDev)
- energencyWithdraw(address _token, address _to, uint256 _amount)

## The `onlyFactory` modifier:

Contract `Linearly`:

- add(address saft, uint256 startTime, uint256 endTime, uint256 count)

Contract `Onetime`:

- add(address saft, uint256 _releaseTime)

Contract `Staged`:

- add(address saft, uint256[] memory _releaseTimes, uint256[] memory _releaseAmounts)

Contract `Saft`:

- mintSaft(address _to, uint256 _lockedAmount)

## The `onlyOwner` modifier:

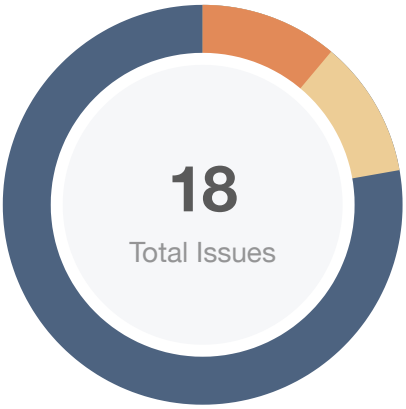
Contract `NFTFund`:

- setPayment(address _payment, bool _status)
- addMarket(address _market)
- removeMarket(address _market)

Contract `SaftFactory`:

- black(address addr)
- setFee(uint256 _fee)
- claimFee(address to)
- addVesting(address addr)

# Findings




| | |
|---|---|
| **Critical** | **0** (0.00%) |
| **Major** | **2** (11.11%) |
| **Medium** | **0** (0.00%) |
| **Minor** | **2** (11.11%) |
| **Informational** | **14** (77.78%) |
| **Discussion** | **0** (0.00%) |

| ID | Title | Category | Severity | Status |
|---|---|---|---|---|
| **GLOBAL-01** | Centralization Related Risks | **Centralization / Privilege** | **Major** | Acknowledged |
| GLOBAL-02 | Missing Emit Events | Coding Style | Informational | Acknowledged |
| GLOBAL-03 | Function Visibility Optimization | Gas Optimization | Informational | Resolved |
| GLOBAL-04 | Unlocked Compiler Version | Language Specific | Informational | Resolved |
| ACP-01 | Missing Input Validation | Volatile Code | Informational | Resolved |
| AKP-01 | Missing Input Validation | Volatile Code | Informational | Resolved |
| BAS-01 | Missing Input Validation | Logical Issue | Minor | Resolved |
| BAS-02 | Missing Input Validation | Logical Issue | Minor | Resolved |
| DUT-01 | Missing Input Validation | Volatile Code | Informational | Resolved |
| NFT-01 | Missing Validation for Array Length | Logical Issue | Informational | Resolved |
| NFT-02 | Unused Variable | Logical Issue | Informational | Resolved |
| SAF-01 | Incorrect Require Condition | Logical Issue | Major | Resolved |
| SAF-02 | Function Name Typo | Coding Style | Informational | Resolved |
| SAF-03 | Unknown Behavior for Default Value | Logical Issue | Informational | Acknowledged |
| SAT-01 | Missing Error Messages | Coding Style | Informational | Resolved |
| SAT-02 | Unused Struct | Logical Issue | Informational | Resolved |

| ID | Title | Category | Severity | Status |
| --- | --- | --- | --- | --- |
| SAT-03 | Unused Variable | Logical Issue | Informational | Resolved |
| SAT-04 | Unused Function `black` | Gas Optimization | Informational | Acknowledged |

# GLOBAL-01 | Centralization Related Risks

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

## Description

In the contract `BaseMarket`, the role `dev` has the authority over the following function:

- setDevAddr()
- setFeeRatio()

In the contract `EnglishAuction`, the role `dev` has the authority over the following function:

- setPriceIncrRatio()

In the contract `Linearly`, the role `factory` has the authority over the following function:

- add()

In the contract `Onetime`, the role `factory` has the authority over the following function:

- add()

In the contract `Staged`, the role `factory` has the authority over the following function:

- add()

In the contract `NFTFund`, the role `owner` has the authority over the following function:

- setPayment()
- addMarket()
- removeMarket()

In the contract `Saft`, the role `dev/factory` has the authority over the following function:

- transferDevAddr()
- energencyWithdraw()
- mintSaft()

In the contract `SaftFactory`, the role `owner` has the authority over the following function:

- black()

- setFee()
- claimFee()
- addVesting()

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

No alleviation.

# GLOBAL-02 | Missing Emit Events

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

## Description

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

**contract `BaseMarket`**

- `__base_init()`
- `setDevAddr()`
- `setFeeRatio()`

**contract `EnglishAuction`**

- `initialize()`
- `setPriceIncrRatio()`

**contract `DeFiAIFarm`**

- `setVestingMaster()`
- `setDevSupply()`

**contract `DeFiAIStratX2`**

- `setBuyBackRate()`

**contract `BaseVesting`**

- `constructor()`

**contract `Saft`**

- `transferDevAddr()`

**contract `SaftFactory`**

- `initialize()`
- `setFee()`

- `claimFee()`
- `addVesting()`

## Recommendation

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

## Alleviation

No alleviation.

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

In the contract `BaseMarket`:

- `setDevAddr()` in L32
- `setFeeRatio()` in L37

In the contract `DutchAuction`:

- `initialize()` in L34
- `createAuction()` in L63
- `cancelAuction()` in L87
- `buy()` in L95

In the contract `EnglishAuction`:

- `initialize()` in L38
- `setPriceIncrRatio()` in L44
- `createAuction()` in L57
- `cancelAuction()` in L81
- `placeBid()` in L102
- `cancelBid()` in L121
- `accept()` in L134

In the contract `FixedPrice`:

- `initialize()` in L38

In the contract `Saft`:

- `transferDevAddr()` in L86

- `energencyWithdraw()` in L94
- `burnSaft()` in L109
- `verify()` in L152

In the contract `SaftFactory`:

- `initialize()` in L45
- `black()` in L56
- `setFee()` in L60
- `claimFee()` in L64
- `addVesting()` in L68
- `createOnetime()` in L115
- `createLinearly()` in L123
- `createStaged()` in L131

## 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 development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# GLOBAL-04 | Unlocked Compiler Version

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Language Specific | Informational | Global | Resolved |

## Description

The contracts have unlocked compiler versions. 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 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

We advise that the compiler version is alternatively locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.0` the contract should contain the following line:

```
pragma solidity 0.8.0;
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# ACP-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | projects/Accessifi/vestings/basevesting.sol (e49cad7): 8 | Resolved |

## Description

There is no valuation to check whether the auction exists.

## Recommendation

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

```
require(_factory != address(0), "_factory can not be zero address.");
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# AKP-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | projects/Accessifi/vestings/linearly.sol (e49cad7): 23 | Resolved |

## Description

There is no valuation to check whether the auction exists.

## Recommendation

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

```
require(startTime < endTime, "startTime must be less than endTime.");
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

## BAS-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | projects/Accessifi/markets/basemarket.sol (e49cad7): 37~38 | Resolved |

### Description

The given input is missing the sanity check.

### Recommendation

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

```
function setFeeRatio(uint256 _feeRatio) public onlyDev {
    require(_feeRatio < 10000, "BaseMarket: invalid _feeRatio");
    feeRatio = _feeRatio;
}
```

### Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# BAS-02 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Minor | projects/Accessifi/markets/basemarket.sol (e49cad7): 15 | Resolved |

## Description

The given input is missing the sanity check.

## Recommendation

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

```
require(_fund != address(0), "_fund can not be zero address.");
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# DUT-01 | Missing Input Validation

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Volatile Code | Informational | projects/Accessifi/markets/dutchauction.sol (e49cad7): 49 | Resolved |

## Description

There is no valuation to check whether the auction exists.

## Recommendation

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

```
function getCurrentPrice(bytes32 _listId) public view returns(uint256) {
    Auction memory auction = auctions[_listId];
    require(auction.owner != address(0), "DutchAuction: auction not exist");
    return _getCurrentPrice(auction);
}
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

## NFT-01 | Missing Validation for Array Length

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | projects/Accessifi/nfttfund.sol (e49cad7): 100, 107, 115 | Resolved |

### Description

There is no validation between `_nfts.length` and `_tokenIds.length` in functions .

### Recommendation

Consider adding validation like below:

```
require(_nfts.length==_tokenIds.length, "NFTFund: _nfts.length and _tokenIds.length are not same");
```

### Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# NFT-02 | Unused Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | projects/Accessifi/nftfund.sol (e49cad7): 16 | Resolved |

## Description

The variable `fees` is declared but never used or updated.

## Recommendation

We recommend removing the unused variable if it is not intended to be used.

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# SAF-01 | Incorrect Require Condition

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Major | projects/Accessifi/saft.sol (e49cad7): 112 | Resolved |

## Description

The condition is not correct, it should be `item.lockedAmount != 0`.

## Recommendation

We advise the client to change the condition as below:

```
112  require(item.lockedAmount != 0, "BaseSaft: invalid tokenId");
```

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# SAF-02 | Function Name Typo

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | projects/Accessifi/saft.sol (e49cad7): 94 | Resolved |

## Description

Function name is mistakenly set as `energencyWithdraw()`.

## Recommendation

We advise the client to fix the typo and set the correct name `emergencyWithdraw()` for the specific function.

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

## SAF-03 | Unknown Behavior for Default Value

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | projects/Accessifi/saft.sol (e49cad7): 154 | Acknowledged |

### Description

We want to know under what circumstances the `verify()` function will be called? In line 55, when `haveToken` is true, the value assigned to `iDocHash` conflicts with the require condition on line 154. When `haveToken` is true, the `verify()` function cannot be called normally.

### Alleviation

The development team responded that only when `haveToken` is false, function `verify()` will be called.

## SAT-01 | Missing Error Messages

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Coding Style | Informational | projects/Accessifi/saftfactory.sol (e49cad7): 65 | Resolved |

### Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

### Recommendation

We advise adding error messages to the linked **require** statements.

### Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

## SAT-02 | Unused Struct

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | projects/Accessifi/saftfactory.sol (e49cad7): 18 | Resolved |

### Description

The struct `TokenCreator` is unused.

### Recommendation

We advise removing the unused struct.

### Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# SAT-03 | Unused Variable

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Logical Issue | Informational | projects/Accessifi/saftfactory.sol (e49cad7): 29 | Resolved |

## Description

The variable `_safts` is declared but never used or updated.

## Recommendation

We recommend removing the unused variable if it is not intended to be used.

## Alleviation

The development team resolved this issue in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

## SAT-04 | Unused Function `black`

| Category | Severity | Location | Status |
| --- | --- | --- | --- |
| Gas Optimization | Informational | projects/Accessifi/saftfactory.sol (e49cad7): 56 | Acknowledged |

### Description

The `black()` function is used to set the blacklist, but the function does not implement the relevant logic.

### Recommendation

We recommend removing the `black()` function or adding related logic.

### Alleviation

The development team has added comments to this function in commit `29e83d9fdbf4660c69176921e7247a2e41c9275e`.

# 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.

### 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.