CERTIK

Security Assessment

Kyoko Ⅲ-p2p

Aug 18th, 2022

Table of Contents

Summary

This report has been prepared for Kyoko Ⅲ-p2p to discover issues and vulnerabilities in the source code of the Kyoko Ⅲ-p2p 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	Kyoko Ⅲ-p2p
Platform	Other
Language	Solidity
Codebase	https://github.com/kyoko-finance/kyoko-p2p-contract
Commit	1. 7d72bff5bc679f163f422b7459e7aa7545022bc4 2. 9f1f61efbb2590eb02e9c93f6bf0341da468e152 3. 7afb98e9ffa0f56e5277aa45366255011f947fa9

Audit Summary

Delivery Date	Aug 18, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
IKP	IKyoko.sol	2434aa2bebe459e9c726d5a0031c2286c9894fe9d02566da47e9c0357e3baead
KSP	KyokoStorage.sol	589a247381a1b4aa21b2e50c857be34e4191e41b87abc80346fe7f9693eeb8c4
DTP	DataTypes.sol	97d0f38259553de11cf732af05918cadb1fab12188365aee84a9aea6e054dfa8
LTP	LenderToken.sol	7ed2ff5a2bb22ea40b6e01b214e3dbb99783e7c3a1004b1d2b16084b972f230c
CPP	Configuration.sol	5e8cc6b4c8eef679d5a44da662c411089673609b7554fceb2693b9e7f0a4b316
KPP	KyokoP2P.sol	9538ccd36e05b9b6aa4ac05fa45486a827509a9bf4c178c5c0f36888f2ee1f2d

Findings

Critical	0	(0.00%)
Major	2	(40.00%)
Medium	1	(20.00%)
Minor	1	(20.00%)
Informational	1	(20.00%)
Discussion	0	(0.00%)

ID	Title	Category	Severity	Status
KPP-01	Centralization Risk In KyokoP2P.Sol	**Centralization / Privilege**	**Major**	Mitigated
KPP-02	Potential Reentrancy Attack	Logical Issue	Major	Resolved
KPP-03	Potential Loss Funds When Modifying Fees	Logical Issue	Medium	Resolved
KPP-04	No Upper Limit For Fee Rate	Logical Issue	Minor	Resolved
PPC-01	Missing Emit Events	Coding Style	Informational	Resolved

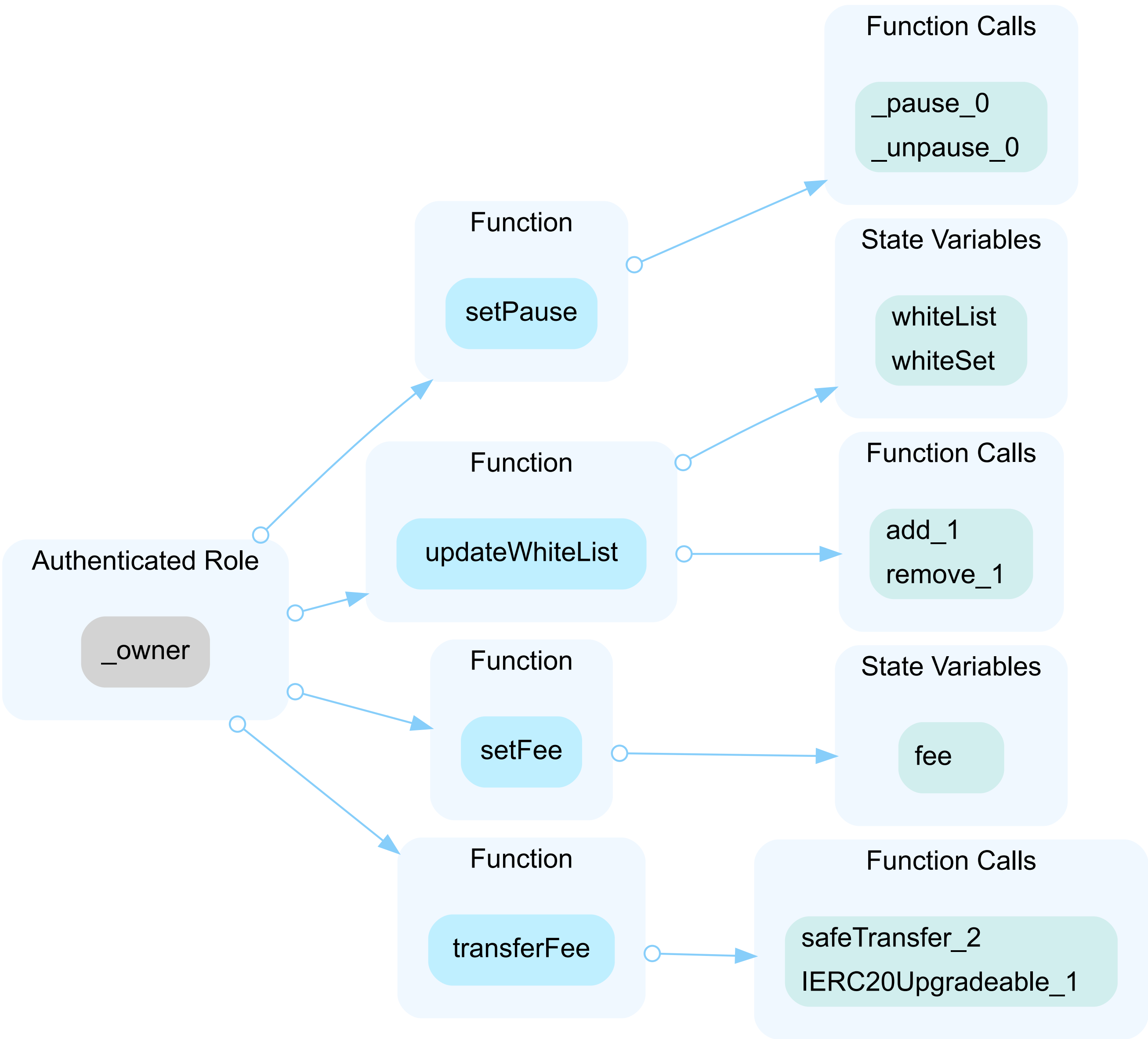
KPP-01 | Centralization Risk In KyokoP2P.Sol

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	KyokoP2P.sol: 57~63, 65~73, 75~78, 540~546	Mitigated

Description

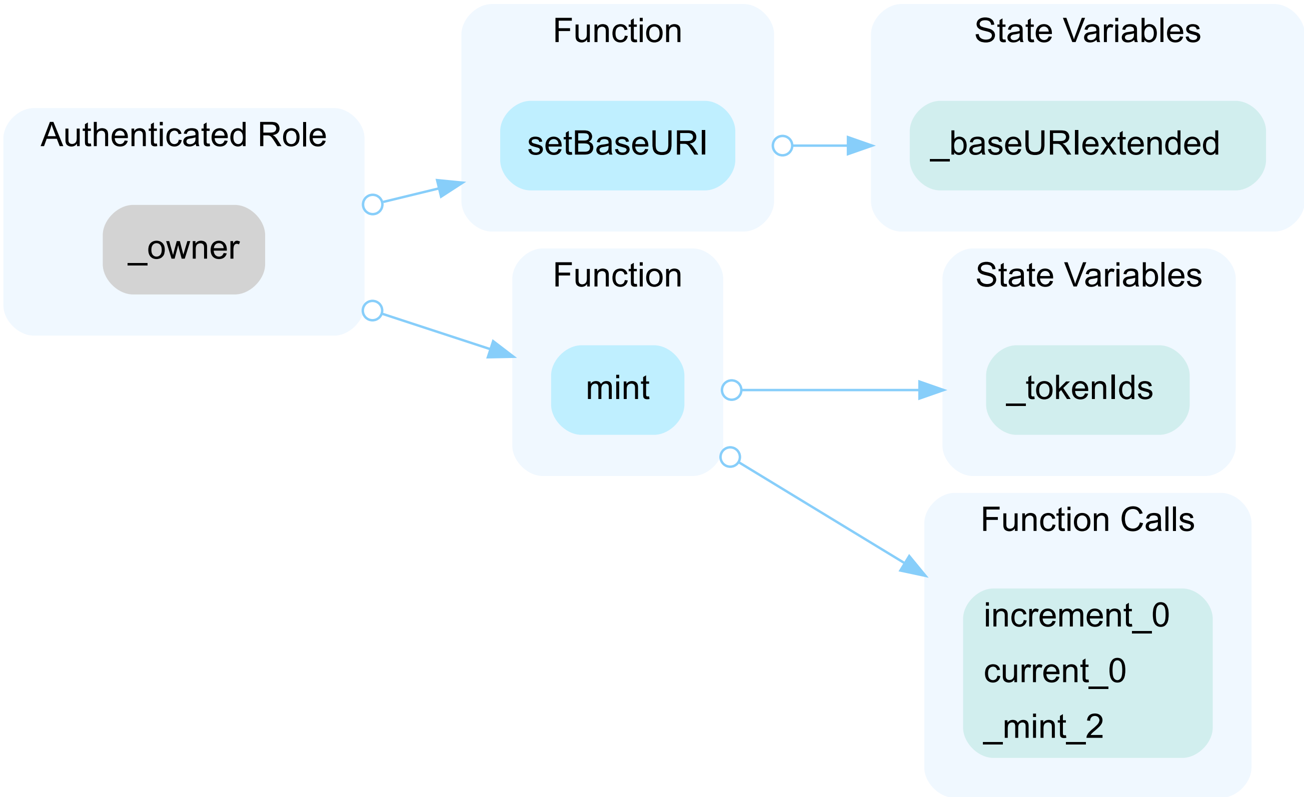
In the contract `KyokoP2P` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and transfer away all the erc20 tokens in the contract.



In the contract `LenderToken` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and change the base URL of the NFT token.



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., multisignature wallets.

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

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Kyoko]:

1. The `owner` account of `KyokoP2P` will be transferred to multi-signature wallets after deployment. The `transferFee()` function has been removed and will be handled through DAO and Timelock later.

2. The `owner` account in `LenderToken` is changed to `AccessControlEnumerableUpgradeable` for management. The mint permission will be authorized to the `KyokoP2P` contract via `script 4`, and the `setBaseURI()` function permission is authorized to another `ROLE_LTOKEN_MANAGER` role.

KPP-02 | Potential Reentrancy Attack

Category	Severity	Location	Status
Logical Issue	● Major	KyokoP2P.sol: 396~401	✓ 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.

1. The attacker deposits his/her NFT token into this contract.
2. The attacker waits for the user to give the offer.
3. When someone calls the `addOffer()` function, the attacker will call `claimCollateral()` after.
4. In the `claimCollateral()`, the `ERC721.SafeTransferFrom()` calls the `msg.sender.onReceived()`.
5. The attacker can reentrancy the contract in the `onReceived()` function.
6. The attacker will call the `acceptOffer()` to get the ERC20 tokens.
7. And then, the `_nft.marks` will be set to `borrowed`.
8. The attacker gets the ERC20 tokens without any collateral.

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

Kyoko team has resolved this issue in commit 0c031f17984235ad7c5a306a7d86e4c9ae3c40ec.

KPP-03 | Potential Loss Funds When Modifying Fees

Category	Severity	Location	Status
Logical Issue	Medium	KyokoP2P.sol: 194~196	Resolved

Description

In the `addOffer()`, the contract calculates the price by deducting the fees. In the `cancelOffer()`, the contract calculates the price by adding the fees. Normally, if the `fee` did not change between the two functions being called, the price will not change. However, if the `fee` changes, the prices before and after will be different. It may cause losses.

Recommendation

Record the amount of tokens the lender really pay may be better.

Alleviation

The Kyoko team added a `_offer.fee` in commit `fb879b9a65b53db739cc92a43b4ccf931e3003c5`.

KPP-04 | No Upper Limit For Fee Rate

Category	Severity	Location	Status
Logical Issue	Minor	KyokoP2P.sol: 76~77	Resolved

Description

In the current implementation, there is no upper limit for the fee rate. Misuse of these fee setting functions could damage the whole protocol. For example, the owner can set the fee rate to more than 100% to cause all transactions to revert.

Recommendation

We recommend setting a reasonable upper limit of `FeeRate`, such as 10% or 20%.

Alleviation

Kyoko team added a cap for `FeeRate` which should be less than 10%, the change was supplied in commit `fb879b9a65b53db739cc92a43b4ccf931e3003c5`.

PPC-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	KyokoP2P.sol: 57~63, 540~546; LenderToken.sol: 34~36, 42~47; test/AssetToken.sol: 33~35	Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The Kyoko team emitted events for the sensitive functions in commit `9f1f61efbb2590eb02e9c93f6bf0341da468e152`.

Appendix

Details on Formal Verification

Technical description

All Solidity smart contracts from the project that implement the ERC-20 interface are in scope of the analysis. Each such contract is compiled into a mathematical model which reflects all possible behaviors of the contract. All subsequent verification results are based on that model, which is designed specifically to be amenable to automated analysis by theorem provers and symbolic model checkers. Apart from representing all possible behaviors of the smart contract, the model also incorporates a verification harness that formalizes the initialization and interaction patterns for the contract. In particular, we use a verification harness that non-deterministically selects a public or external function and models its execution. The contract state is initialized non-deterministically (i.e. by arbitrary values) before invocation of the function. Hence, the mathematical model over-approximates the reachable state space of the contract throughout any actual deployment on chain. By doing so, all verification results carry over to the contract's behavior in arbitrary states after it has been deployed. Once the model is constructed, our analysis engine attempts to prove that all executions of the contract are subsumed by a set of pre-defined specifications which capture the desired and admissible behaviors of the smart contract. For the scope of this audit, we use 38 property specifications that cover the functionality of the functions as stated in Sec. Scope.

Assumptions and simplifications

The following assumptions and simplifications have been applied during formal verification:

- Gas consumption is not taken into account, i.e. we assume that executions do not terminate prematurely because they run out of gas.
- The contract's state variables are non-deterministically initialized before invocation of any of those functions. That ignores contract invariants and may lead to false positives. It is, however, a safe over-approximation.
- The verification engine reasons about unbounded integers. Machine arithmetic is modeled as operations on the congruence classes arising from the bit-width of the underlying numeric type. This ensures that over- and underflow characteristics are faithfully represented.

Formalism for property definitions

This section provides details on the 38 formal specifications that were in scope of the audit. All properties are expressed in linear temporal logic (LTL). In that context, we consider all invocations and returns from public and external functions as discrete time steps. Thus, our analysis reasons about the contract's state upon entering and leaving public and external functions.

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `started(f, [cond])` Indicates an invocation of contract function `f` within a state satisfying formula `cond`.
- `willSucceed(f, [cond])` Indicates an invocation of contract function `f` within a state satisfying formula `cond` and considers only those executions that do not revert.
- `finished(f, [cond])` Indicates that execution returns from contract function `f` in a state satisfying formula `cond`. Here, formula `cond` may refer to the contract's state variables and to the value they had upon entering the function (using the `old` function).
- `reverted(f, [cond])` Indicates that execution of contract function `f` was interrupted by an exception in a contract state satisfying formula `cond`.

The verification performed in this audit is restricted to pre- and postconditions of procedure invocations. The used model consists of a harness that invokes a non-deterministically selected function of the contract's public and external interface. All formulas are analyzed w.r.t. the trace that corresponds to this function invocation.

Properties for ERC-20 function `transfer(to, amount)`

erc20-transfer-correct-amount

It is expected that non-reverting invocations of `transfer(recipient, amount)` that return `true` subtract the value in `amount` from the balance of the address `msg.sender` and add the same value to the balance entry of the `recipient` address.

```
[](willSucceed(transfer(to, value), to != msg.sender)
   ==> <> (finished(transfer(to, value),
                 return == true
                 ==> balance[msg.sender] == old(balance[msg.sender]) - value
                     && balance[to] == old(balance[to]) + value)))
```

erc20-transfer-correct-amount-self

It is expected that non-reverting invocations of `transfer(recipient, amount)` that return `true` and where the address in `recipient` equals the address of `msg.sender` (i.e. self-transfers) do not change the balance of address `msg.sender`.

```
[](willSucceed(transfer(to, value), to == msg.sender)
   ==> <> (finished(transfer(to, value),
                 return == true
                 ==> balance[to] == old(balance[to]))))
```

Properties for ERC-20 function `transferFrom(from, to, amount)`

erc20-transferfrom-revert-from-zero

It is expected that calls of the form `transferFrom(from, dest, amount)` fail if the address value provided in the `from` in-parameter is the zero address.

```
[](started(transferFrom(from, to, value), from == 0)
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-revert-to-zero

It is expected that calls of the form `transferFrom(from, dest, amount)` fail if the address value provided in the `dest` in-parameter is the zero address.

```
[](started(transferFrom(from, to, value), to == 0)
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-fail-exceed-balance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the balance of address `from` is expected to fail.

```
[](started(transferFrom(from, to, value), value > balance[from])
   ==> <> (reverted(transferFrom) || finished(transferFrom, return == false)))
```

erc20-transferfrom-correct-allowance

It is expected that non-reverting invocations of `transferFrom(from, to, amount)` that return `true` decrease the allowance of the address in `msg.sender` for the address in `from` by the value in `amount`. Two special cases are taken into account:

1. An allowance that equals `type(uint256).max` is treated as an exception and interpreted as an unlimited allowance that does not need to be reduced in order for this check to pass.
2. If the owner of the tokens that are transferred invokes `transferFrom` (i.e. when the address in `msg.sender` equals the address in `from`) we do not require an update of the allowance.

```
[](willSucceed(transferFrom(from, to, value))
   ==> <> finished(transferFrom(from, to, value),
                  return == true
                  ==> ((allowances[from][msg.sender] == old(allowances[from][msg.sender]) -
value)
                       || (allowances[from][msg.sender] == old(allowances[from][msg.sender])
                           && (from == msg.sender
                               || old(allowances[from][msg.sender]) == type(uint256).max)))))
```

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.

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.

CERTIK