ILLUST SPACE

SMART CONTRACT AUDIT

ZOKYO.

Aug 6th, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

SCORE

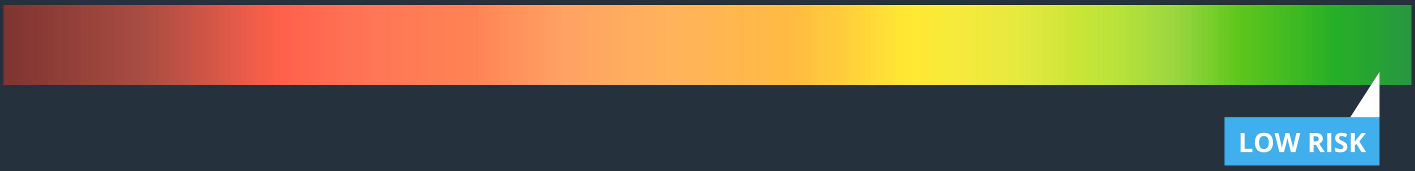
100

TECHNICAL SUMMARY

This document outlines the overall security of the Illust Space smart contracts, evaluated by Zokyo's Blockchain Security team.

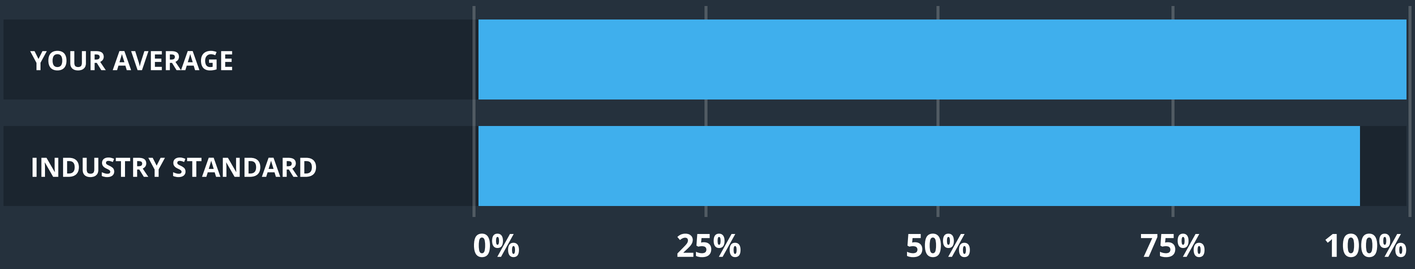
The scope of this audit was to analyze and document the Illust Space smart contract codebase for quality, security, and correctness.

Contract Status



There were no critical issues found during the audit.

Testable Code



The testable code is 100%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Illust Space team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the Illust Space repository – https://github.com/illustspace/illust-contracts/commit/06d04db873ca25a31508afc0b2dd39fc40b63f8a

Requirements:

- https://github.com/illustspace/illust-contracts

Contracts:

- Ainsoph.sol
- IllustMarketplace.sol

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.

SUMMARY

Zokyo auditing team has run a deep investigation of Illust Space's smart contracts. During the auditing process there were no issues found. The contracts are in excellent condition. They are well written and structured.

Based on the conducted audit, we give a score of 100 to the aforementioned contracts. Zokyo auditing team can state that the contracts are fully production ready and bear no security or operational risk.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● Low

The issue has minimal impact on the contract’s ability to operate.

● Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

During the auditing process (both manual part and testing part) no issues were identified.

	Ainsoph.sol	IllustMarketplace.sol
Re-entrancy	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected
Unexpected Ether	Not affected	Not affected
Delegatecall	Not affected	Not affected
Default Public Visibility	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected
External Contract Referencing	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected
Race Conditions / Front Running	Not affected	Not affected
General Denial Of Service (DOS)	Not affected	Not affected
Uninitialized Storage Pointers	Not affected	Not affected
Floating Points and Precision	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected
Signatures Replay	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Illust Space team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	91.96	54.00	77.27	92.04	
Ainsoph.sol	94.14	65.00	85.71	94.12	... 295, 300, 341
IllustMarketplace.sol	90.16	46.67	62.50	90.32	... 262, 273, 277
All files	**91.96**	**54.00**	**77.27**	**92.04**	

Test Results

```
IllustCustody
  mintAsset
    ✓ cannot mint an asset when not whitelisted (76ms)
    ✓ lets an admin mint assets (88ms)
    ✓ mints an asset when whitelisted (72ms)
    ✓ cannot mint an asset that has already been minted (100ms)
  royaltyInfo
    ✓ calculates the total royalties (75ms)
  verifyRoyalties
    ✓ verifies royalties that match the saved ones (79ms)
    ✓ verifies royalties that exceed the saved ones (77ms)
    ✓ fails verification for missing royalties receivers (86ms)
    ✓ fails verification for royalties that are less than the saved ones (78ms)
  distributeRoyalties
    ✓ distributes royalties (114ms)
    ✓ distributes royalties again (114ms)
  pausing
```

```
      ✓ cannot mint when paused (69ms)
      ✓ can mint after un-pausing (126ms)
    transfer
      ✓ cannot transfer (69ms)
      ✓ can do marketplace transfer for approved marketplace (135ms)
      ✓ can self-transfer when open trading is allowed (135ms)
    interfaces
      ✓ supports ERC2981
      ✓ supports ERC721

  IllustCustody
    trading
      ✓ does a trade (406ms)

  19 passing (6s)
```

Tests written by Zokyo Security team

As part of our work assisting Illust Space in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Illust Space contract requirements for details about issuance amounts and how the system handles these.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	100.00	100.00	100.00	100.00	
Ainsoph.sol	100.00	100.00	100.00	100.00	
IllustMarketplace.sol	100.00	100.00	100.00	100.00	
All files	**100.00**	**100.00**	**100.00**	**100.00**	

Test Results

```
Contract: Ainsoph
  Ainsoph Deploy Phase Test Cases
    ✓ should deploy with correct minter (172ms)
    ✓ should deploy with correct marketplace fee address (88ms)
    ✓ should deploy with correct marketplace fee percentage (99ms)
    ✓ should deploy with correct name (103ms)
    ✓ should deploy with correct symbol (83ms)
    ✓ should deploy with correct supported interfaces (178ms)
  Ainsoph Asset Phase Test Cases
    ✓ should mint asset correctly (681ms)
    ✓ should create correct tokenURI (104ms)
    ✓ should change base URI correctly (293ms)
    ✓ shouldn't change base URI by not the admin (801ms)
```

```
        ✓ shouldn't mint by not the minter (229ms)
        ✓ shouldn't mint with royalties greater than 100% (503ms)
        ✓ shouldn't mint if token already minted (740ms)
        ✓ should return correct royalty info (94ms)
        ✓ should verify royalties correctly (118ms)
        ✓ shouldn't verify royalties with to low percentage (110ms)
        ✓ shouldn't verify royalties with missing royalty receiver (93ms)
        ✓ shouldn't verify royalties with to low marketplace fee percentage (74ms)
        ✓ should distribute royalties correctly (362ms)
        ✓ shouldn't set open trading by not the admin (210ms)
        ✓ should set open trading correctly (840ms)
        ✓ shouldn't pause by not the admin (188ms)
        ✓ should pause correctly (914ms)
        ✓ shouldn't unpause by not the admin (166ms)
        ✓ should unpause correctly (304ms)

  Contract: IllustMarketplace
    IllustMarketplace Test Cases
        ✓ should deploy with correct token contract (94ms)
        ✓ should deploy with correct marketplace fee address (82ms)
        ✓ shouldn't set royalties by not the admin (166ms)
        ✓ shouldn't set royalties greater than 100% (565ms)
        ✓ should set royalties correctly (465ms)
        ✓ shouldn't accept bid by not the owner of asset (179ms)
        ✓ shouldn't accept bid if seller not approve transfers (175ms)
        ✓ should accept bid correctly (670ms)
        ✓ should accept payment for first sale of asset correctly (791ms)
        ✓ should accept payment for subsequent sales of asset correctly (1319ms)
        ✓ shouldn't accept payment for asset if seller didn't accept bid (210ms)
        ✓ shouldn't accept payment for auction that is ended (161ms)
        ✓ shouldn't accept payment from not the winning address (219ms)
        ✓ shouldn't accept payment if pay amount is to low (190ms)
        ✓ shouldn't accept payment if asset owner is a winner (201ms)
        ✓ shouldn't set ainsoph contract by not the admin (194ms)
        ✓ should set ainsoph contract correctly (353ms)
        ✓ should receive funds correctly (206ms)
        ✓ shouldn't withdraw funds by not the admin (162ms)
        ✓ should withdraw funds correctly (222ms)
```

```
      ✓ shouldn't pause by not the admin (179ms)
      ✓ should pause correctly (921ms)
      ✓ shouldn't unpause by not the admin (195ms)
      ✓ should unpause correctly (246ms)


  49 passing (23s)
```

We are grateful to have been given the opportunity to work with the Illust Space team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the Illust Space team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.