ONYX LABS

Smart Contract Audit

Prepared For:
Infinit3 Studios
Jira Group

Date:
7/27/22

Table of Contents

Summary

This report was prepared to summarize the findings of the audit performed for Infinit3 Studios/Jira Group of the Jiraverse ERC-721 minting contract, on July 27th, 2022. The primary objective of the audit was to identify issues and vulnerabilities in the source code. Due to the short turn-around time, this examination was performed primarily utilizing Manual Review techniques. Static analysis and other testing methods were not performed by the audit team.

The audit consisted of:

- 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.
- Line-by-line manual review of the entire codebase by auditors.

The following report contains the results of the audit, and recommendations to guard against potential security threats and improve contract functionality.

We would like to thank the Infinit3 Studios team for their business and cooperation; their openness and communication made this audit a success.

Overview

Project Summary

Project Name	Jiraverse
Description	Jiraverse is an ERC721 NFT mint, consisting of two phases. Phase 1 includes a public fair dutch auction, and Phase 2 includes a merkle tree based claim/mint.
Blockchain	Ethereum
Language	Solidity
Codebase	Jiraverse.sol - Provided Directly
Commit	N/A

Audit Summary

Delivery Date	7/27/22
Audit Methodology	Manual Review
Key Components	N/A

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Resolved
Critical	0	0	0	0	0
High	2	0	0	1	1
Medium	5	0	0	1	4
Low	1	0	0	0	1
Informational	4	0	0	3	1

Audit Scope

ID	File
JIRA	Jiraverse.sol

Findings

ID	Title	Category	Severity	Status
JIRA-001	No method to refund DA	Security	High	Resolved
JIRA-002	Funds accessible before refund	Security	High	Acknowledged
JIRA-003	Transfer vs Call	Security	Medium	Acknowledged
JIRA-004	No Symbol in constructor	Functional	Low	Resolved
JIRA-005	Unused Function	Functional	Information	Resolved
JIRA-006	<=/>= Usage	Optimization	Low	Resolved
JIRA-007	Use of Constants	Optimization	Medium	Resolved
JIRA-008	Use timestamp for public mint	Optimization	Low	Resolved
JIRA-009	Use setAux()	Optimization	Medium	Resolved
JIRA-010	Variable Naming	Style	Information	Acknowledged
JIRA-011	Declaring variables	Style	Information	Acknowledged
JIRA-012	Declare interface	Style	Information	Acknowledged

JIRA-001 | No Method to Refund Dutch Auction

Category	Severity	Location	Status
Security	High	N/A	Resolved

Description

The current contract does not provide an on-chain method of refunding users who pay less than the final Dutch Auction price. While this action can be accomplished off-chain, it is advisable to complete the action on-chain via a claim or automatic dispersal.

Without completing this action on-chain, customers will not have any transparency into what funds are accessible, or when/how they will receive their funds. Building in a trust-less refund function will improve the overall security of the process, and decrease the likelihood of customer complaints/legal.

Recommendation

Incorporate a refund claim function to allow users to claim the funds due.

JIRA-002 | Funds Withdrawable before Refund

Category	Severity	Location	Status
Security	High	Line 166	Acknowledged

Description

IF the contract is modified to include on-chain refund mechanics, a method for securing said funds in the contract for a period of time should also be included. If it is not, then the JG team cannot access their funds without removing the funds required to refund customers. This scenario would be detrimental to both the customers and Infinit3 Studios/JG.

Recommendation

Modify the withdraw function to only access the funds due to the project, and lock any refund funds in the contract.

JIRA-003 | Transfer Used in Withdraw Function

Category	Severity	Location	Status
Security	Medium	Line 166	Acknowledged

Description

The current withdraw function includes the use of payable transfer. While this can be an effective function, it limits the flexibility and can prevent ETH from being withdrawn to certain addresses due to inherent gas limitations with payable transfer. The contract may have issues sending funds to multi signature contracts.

Using payable call as a method of sending ETH to an address is ideal, as it does not

Recommendation

Replace payable transfer with payable call.

Example

```
Replace:
   function withdrawETH(address _to) external onlyOwner {
       payable(_to).transfer(address(this).balance);
   }

With:
   function withdrawETH(address _to) external onlyOwner {
       (bool success, ) = _to.call{value: address(this).balance}("");
       require(success, "Transfer failed.");
   }
```

JIRA-004 | No Symbol in Constructor

Category	Severity	Location	Status
Functional	Low	Line 42	Resolved

Description

The ERC721A Constructor used in the contract contains a blank field where the contract symbol should reside. Leaving this defined as a blank string could cause confusion, and should be replaced with a recognizable symbol string.

Recommendation

Replace the blank string in the constructor argument with a recognizable symbol, such as “JIRA”.

Example

```
Replace:
   constructor () ERC721A("Jiraverse", "") { }

With:
   constructor () ERC721A("Jiraverse", "JIRA") { }
```

JIRA-005 | Unused Function

Category	Severity	Location	Status
Functional	Information	Line 67	Resolved

Description

The function ‘getNumClaimed()’ was declared but not used. It should be removed.

Recommendation

Removed the unused function from the contract.

Example

```
Remove:
function getNumClaimed(address _user) public view returns(uint64) {
       return _getAux(_user);
   }
```

JIRA-006 | <=/>= Usage

Category	Severity	Location	Status
Optimization	Low	Several	Resolved

Description

The contract frequently utilizes the >= or <= operator. This is acceptable in situations where necessary, however if > or < can be utilized by modifying a variable (i.e. require < 5 instead of <= 4) gas saving can be achieved.

Recommendation

Replace >=/<= with </> in lines 42, 84, 99, 114, 117, and 144.

Example

```
Replace:
   require(_totalMinted() + _quantity <= 6001, "Not enough tokens left to mint");
With:
   require(_totalMinted() + _quantity < 6002, "Not enough tokens left to mint");
```

JIRA-007 | Use of Constants

Category	Severity	Location	Status
Optimization	Medium	Several	Resolved

Description

Where possible, replacing stored variables with constant values can provide significant gas savings. This should be done when you know the value and there is no possibility of it changing (i.e. a known contract address).

Recommendation

Replace stored values time_until_decrease, decrease_amount, max_price, min_price, time_to_start_decreasing with constants.

Example

```
Replace:
   uint256 time_until_decrease = 15 minutes;
   uint256 decrease_amount = 0.025 ether;
   uint256 max_price = 0.55 ether;
   uint256 min_price = 0;
   uint256 time_to_start_decreasing = 30 minutes;

With:
   uint256 public constant TIME_UNTIL_DECREASE = 15 minutes;
   uint256 public constant DECREASE_AMOUNT = 0.025 ether;
   uint256 public constant MAX_PRICE = 0.55 ether;
   uint256 public constant MIN_PRICE;
   uint256 public constant TIME_TO_START_DECREASING = 30 minutes;
```

JIRA-008 | Use timestamp for public mint

Category	Severity	Location	Status
Optimization	Low	113	Resolved

Description

Using a timestamp comparison to control access to a function such as public minting can provide gas savings.

Recommendation

Replace the boolean in the public mint function with a timestamp.

Example

```
Replace:
   require(public_sale_running, "Public sale is not running");

With:
   require(block.timestamp > publicSaleStartTime, "Public sale is not running");
```

JIRA-010 | Variable Naming

Category	Severity	Location	Status
Style	Information	Several	Acknowledged

Description

Solidity style guidelines call for the use of mixedCase syntax for local and state variable names. The contract uses lowercase_with_underscore syntax.

Recommendation

Rename local and state variable names to comply with mixedCase syntax.

Example

```
Replace:
   uint public public_sale_start_time;
   mapping(uint => bytes32) public gen1_merkle_roots;

With:
   uint public publicSaleStartTime;
   mapping(uint => bytes32) public gen1MerkleRoots;
```

JIRA-011 | Declaring Variables

Category	Severity	Location	Status
Style	Information	Several	Resolved

Description

The solidity compiler allows for variables to remain undeclared with default values (i.e. 0). This should be done when possible.

Recommendation

Remove declared default values for private_sale_running, public_sale_running, and base_uri.

Example

```
Replace:
   bool public private_sale_running = false;
   bool public public_sale_running = false;

With:
   bool public private_sale_running;
   bool public public_sale_running;
```

JIRA-012 | Set contract instance

Category	Severity	Location	Status
Style	Information	Line 28	Acknowledged

Description

When consistently accessing an external contract throughout the contract, it is best practice to store it as a named instance instead of repeatedly accessing it via an interface.

Recommendation

Replace the JIRA contract address with a named instance of the JIRA contract.

Example

```
Replace:
   address public constant JIRA_TOKEN_ADDRESS = 0x517AB044bda9629E785657DbbCae95C40C8f452C;
   &&
   IERC20(JIRA_TOKEN_ADDRESS).transferFrom(...);

With:
   IERC20 public constant JIRA = IERC20(0x517AB044bda9629E785657DbbCae95C40C8f452C);
   &&
   JIRA.transferFrom(...);
```

Disclaimer

The audit performed by 0nyX Labs is intended to identify function weaknesses, potential security threats, and improve performance. It does not provide a guarantee of contract performance or the absence of any exploitable vulnerabilities. Any alterations to the source code provided could potentially invalidate the analysis provided in this audit report. 0nyX Labs is not liable for any losses or damages incurred as a result of contract deployment and the proceeding occurrences.

About

0nyx Labs is a full-service web3 development firm founded in December 2021. Their scope of expertise ranges from smart contract development for a multitude of applications, to web development and audits. By offering best-in-class service, 0nyx Labs has helped many successful projects enter the web3 space, or provided assistance with their ongoing development needs.