26th MARCH 2021

SMART CONTRACT AUDIT REPORT

-

version v2.0

Smart Contract Security Audit and General Analysis

HAECHI AUDIT

Table of Contents

5 Issues (2 Critical, 2 Major, 1 Minor) Found

About HAECHI AUDIT

HAECHI AUDIT is a global leading smart contract security audit and development firm operated by HAECHI LABS. HAECHI AUDIT consists of professionals with years of experience in blockchain R&D and provides the most reliable smart contract security audit and development services.

So far, based on the HAECHI AUDIT's security audit report, our clients have been successfully listed on the global cryptocurrency exchanges such as Huobi, Upbit, OKEX, and others.

Our notable portfolios include SK Telecom, Ground X by Kakao, and Carry Protocol while HAECHI AUDIT has conducted security audits for the world's top projects and enterprises.

Trusted by the industry leaders, we have been incubated by Samsung Electronics and awarded the Ethereum Foundation Grants and Ethereum Community Fund.

Contact : audit@haechi.io
Website : audit.haechi.io

01. Introduction

This report was written to provide a security audit for the Fractional smart contract. HAECHI AUDIT conducted the audit focusing on whether Fractional smart contract is designed and implemented in accordance with publicly released information and whether it has any security vulnerabilities.

The issues found are classified as CRITICAL, MAJOR, MINOR or TIPS according to their severity.

CRITICAL	Critical issues are security vulnerabilities that MUST be addressed in order to prevent widespread and massive damage.
MAJOR	Major issues contain security vulnerabilities or have faulty implementation issues and need to be fixed.
MINOR	Minor issues are some potential risks that require some degree of modification.
TIPS	Tips could help improve the code's usability and efficiency

HAECHI AUDIT advises addressing all the issues found in this report.

02. Summary

The code used for the audit can be found at GitHub (https://github.com/fractional-company/contracts). The last commit for the code audited is at "37f9428fe9a5596c93ed9dfccb7be41afc0646ee".

Issues

HAECHI AUDIT has 2 Critical Issue, 2 Major Issues, and 1 Minor Issues; 2 TIPS has been included to improve the code quality

Update

[v2.0] - new commit 3a1643789aab257c006198677cbe78ea424899ae has fixed 2 Critical issues, 2 Major issues and applied 1 TIPS. 1 Minor issue and 1 TIPS has been updated.

Severity	Issue	Status
CRITICAL	DoS can happen when the winning address is a contract address.	(Found v1.0) (Resolved v2.0)
CRITICAL	updateUserPrice can set reservePrice out of range.	(Found v1.0) (Resolved v2.0)
MAJOR	Redeem function should send ETH to the winning address	(Found v1.0) (Resolved v2.0)
MAJOR	Can end auction before it starts	(Found v1.0) (Resolved v2.0)
MINOR	claimFee() function will mint tokens eternally and it will mint different fees based on called time.	(Found v1.0) (Updated v2.0)
TIPS	updateAuctionLength/updateBasePrice won't have any effect after auction starts	(Found v1.0) (Updated v2.0)
TIPS	IndexERC721 has unused variables	(Found v1.0) (Resolved v2.0)

03. Overview

Contracts Subject to Audit

- ERC721TokenVault.sol
- ERC721VaultFactory.sol
- IndexERC721.sol
- Settings.sol

04. Issues Found

CRITICAL : DoS can happen when the winning address is a contract address. (Found - v1.0)(Resolved v2.0)

CRITICAL

Problem Statement

ERC721TokenVault#bid() will send eth to the winning bidder if the current bid is successful. But sending eth can execute fallback function. If fallback function is declared to run an infinite loop, bid() function will always fail. This will result in DoS to auction system.

Recommendation

Change sendETHOrWETH function to send WETH as default or send WETH when recipient is contract address. It can also be achieved by stipening the gas of <address>.call function.

Update

[v2.0] fractional team has applied the recommendation to fix this issue

CRITICAL : updateUserPrice can set reservePrice out of range. (Found - v1.0)(Resolved v2.0)

CRITICAL

Problem Statement

ERC721TokenVault#updateUserPrice() is for setting the reservePrice. But unlike _beforeTokenTransfer() function, updateUserPrice() function does not check if reservePrice is between acceptable range.

Recommendation

Check reservePrice is within range when the user calls updateUserPrice().

Update

[v2.0] fractional team has applied the recommendation to fix this issue

MAJOR : Redeem function should send ETH to the winning address. (Found - v1.0)(Resolved v2.0)

MAJOR

Problem Statement

ERC721TokenVault#redeem() is used when someone buys out all the minted tokens. If this happens, bidding will end. But the winning bidder won't receive it's bidded eth.

Recommendation

Change the redeem() function to send eth to the winning bidder.

Update

[v2.0] fractional team has applied the recommendation to fix this issue

MAJOR : Can end auction before it starts. (Found - v1.0)(Resolved v2.0)

MAJOR

Problem Statement

ERC721TokenVault#end() does not check if the auction has started. If auction ends before even it starts, current winning bidder will be zero address so it will result in losing control of the nft

Recommendation

Check if auction has started

+ HAECHI AUDIT recommends the fractional team to change the state from 2 boolean variables to a single enum variable. By applying this method, it will be gas efficient and give more clarity to the code.

Update

[v2.0] fractional team has applied the recommendation to fix this issue

MINOR : claimFee() function will mint tokens eternally and it will mint different fees based on called time. (Found - v1.0)(Updated - v2.0)

MINOR

Problem Statement

ERC721TokenVault#claimFee() will mint tokens even after the auction ends. Also, the number of minted token amounts is not consistent. For example, if the fee percentage of the auction is 2%, totalSupply was 1000 and price the winner paid is 1 eth. curator and governance should both receive 20 tokens which is worth 20/1040 * 1 ETH. But, if someone front runs and executes cash() function prior to claimFee() function to burn 500 tokens and receives 0.5 eth, governance and curator will receive 10 tokens each which is worth 10/520 * 0.5 eth.

Recommendation

Make sure the fee is minted before someone interacts with the contract. And clarify the fee minting period.

Update

[v2.0] fractional team has applied the recommendation but HAECHI has found that this can also lead to DoS for redeem() function

TIPS : updateAuctionLength/updateBasePrice won't have any effect after auction starts (Found - v1.0)(Updated - v2.0)

TIPS

Problem Statement

ERC721TokenVault#updateAuctionLength()/updateBasePrice() function is for setting the auction metadata before auction has started. Although these functions can be called even after the auction has started, it won't have any effect on the auction's metadata.

Recommendation

Revert when these functions are called after the auction has gone live, or change the functions to take effect even.

Update

[v2.0] updateBasePrice has been updated but updateAuctionLength issue still exists.

TIPS : IndexERC721 has unused variables (Found - v1.0)(Resolved - v2.0)

TIPS

Problem Statement

IndexERC721 has two unused variables, _tokenIdTracker and tokenToIndexOwner.

Recommendation

Make sure things are fully implemented and remove the unused variables after that.

Update

[v2.0] fractional team has applied the recommendation to fix this issue

05. Disclaimer

This report is not an advice on investment, nor does it guarantee adequacy of a business model and/or a bug-free code. This report should be used only to discuss known technical problems. The code may include problems on Ethereum that are not included in this report. It will be necessary to resolve addressed issues and conduct thorough tests to ensure the safety of the smart contract.