



VIRTUAL CLAIM

# NICHO-NFT

## Smart Contract Review

Deliverable: Smart Contract Audit Report

Security Assessment  
April 2022

## Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Company. The content, conclusions, and recommendations set out in this publication are elaborated in the specific for only project.

Virtual Caim does not guarantee the authenticity of the project or organization or team of members that are connected/owner behind the project nor the accuracy of the data included in this study. All representations, warranties, undertakings, and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Company nor any person acting on the Company's behalf may be held responsible for the use that may be made of the information contained herein.

Virtual Caim retains the right to display audit reports and other content elements as examples of their work in their portfolio and as content features in other projects with protecting all security purposes of customers. The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

©Virtual Caim Private Limited, 2022.

# Smart Contract Audit

## Report Summary

Title	Nicho-NFT Smart Contract Audit		
Project Owner	Nicho-NFT		
Classification	Public		
Reviewed by	Virtual Caim Private Limited	Review date	28/04/2022
Approved by	Virtual Caim Private Limited	Approval date	28/04/2022
	Nº Pages	25	

# Smart Contract Audit

## Overview

### Background

Nicho-NFT's team requested Virtual Caim to perform an Extensive Smart Contract Audit of their 'NichoAuction v2' Smart Contract.

### Project Dates

The following is the project schedule for this review and report:

- April 28: Smart Contract Review Started (*Completed*)
- April 28: Initial Delivery of Audit Findings (*Completed*)

## Coverage

### Target Specification and Revision

For this audit, we performed project's basic research, investigation by discussing the details with the project owner/developers, and then review the smart contract of Nicho-NFT.

The following documentation & repositories were considered in -scope for the review:

<i>Nicho-NFT Project</i>	<a href="https://github.com/NichoNFT/Contracts/tree/main/audit%20contracts">https://github.com/NichoNFT/Contracts/tree/main/audit%20contracts</a>
--------------------------	---

# Smart Contract Audit

## Introduction

Given the opportunity to review Nicho-NFT's Contract related smart contract source code, we in the report summary our methodical approach to evaluate all potential common security issues in the smart contract implementation, expose possible semantic irregularities between smart contract code and design document, and provide additional suggestions or recommendations for improvement. Our results show that the given version of smart contracts is ready to use after resolving the mentioned issues and done functional testing by owner/developer themselves, as there might be issues related to business logic, security or performance which only can found/understand by them.

## About Audit

Item	Description
<b>Issuer</b>	Nicho-NFT
<b>Website</b>	<a href="https://nichonft.com/">https://nichonft.com/</a>
<b>Type</b>	-
<b>Platform</b>	-
<b>Language</b>	Solidity
<b>Audit Test Method</b>	Whitebox Testing
<b>Latest Audit Report</b>	April 28, 2022

# Smart Contract Audit

## Test Methods Information

### Test Methods

#### Black-box testing

Conduct security tests from an attacker's perspective externally.

#### Grey-box testing

Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.

#### White-box testing

Based on the open-source code, non-open-source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

# Smart Contract Audit

## Vulnerability Severity Level Information

Level	Description
Critical	Critical severity vulnerabilities will have a significant effect on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

# Smart Contract Audit

## List of Check Items

### Basic Coding Bugs

Constructor Mismatch

Ownership Takeover

Redundant Fallback Function

Overflows & Underflows

Reentrancy

MONEY-Giving Bug

Blackhole

Unauthorized Self-Destruct

Revert DoS

Unchecked External Call

Gasless Send

Send Instead of Transfer

Costly Loop

(Unsafe) Use of Untrusted Libraries

(Unsafe) Use of Predictable Variables

Transaction Ordering Dependence

Deprecated Uses

### Advanced DeFi Scrutiny

Business Logic Review

Functionality Checks

Authentication Management

Access Control & Authorization

Oracle Security

Digital Asset Escrow

Kill-Switch Mechanism

Operation Trails & Event Generation

ERC20 Idiosyncrasies Handling

Frontend-Contract Integration

Deployment Consistency

Holistic Risk Management

### Semantic Consistency Checks

Semantic Consistency Checks

### Additional Recommendations

Avoiding Use of Variadic Byte Array

Using Fixed Compiler Version

Making Visibility Level Explicit

Making Type Inference Explicit

Adhering To Function Declaration Strictly

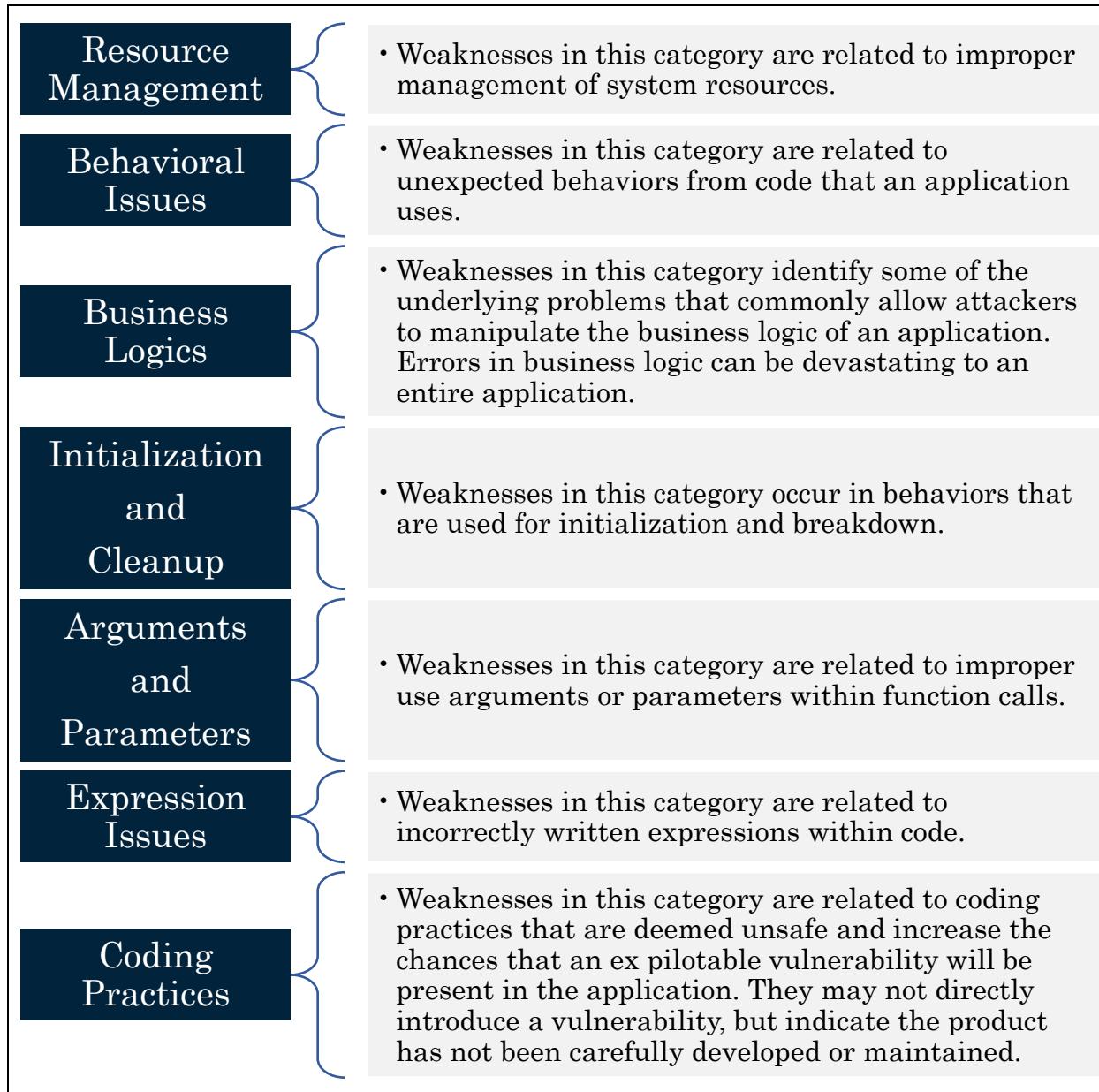
Following Other Best Practices

# Smart Contract Audit

## Common Weakness Enumeration (CWE) Classifications Used in this Audit

Configuration	<ul style="list-style-type: none"><li>Weaknesses in this category are typically introduced during the configuration of the software.</li></ul>
Data Processing Issues	<ul style="list-style-type: none"><li>Weaknesses in this category are typically found in functionality that processes data.</li></ul>
Numeric Errors	<ul style="list-style-type: none"><li>Weaknesses in this category are related to improper calculation or conversion of numbers.</li></ul>
Security Features	<ul style="list-style-type: none"><li>Weaknesses in this category are concerned with topics like authentication, access control, confidentiality, cryptography, and privilege management. (Software security is not security software.)</li></ul>
Time and State	<ul style="list-style-type: none"><li>Weaknesses in this category are related to the improper management of time and state in an environment that supports simultaneous or near-simultaneous computation by multiple systems, processes, or threads.</li></ul>
Error Conditions, Return Values, Status Codes	<ul style="list-style-type: none"><li>Weaknesses in this category include weaknesses that occur if a function does not generate the correct return/status code, or if the application does not handle all possible return/status codes that could be generated by a function.</li></ul>

# Smart Contract Audit



# Smart Contract Audit

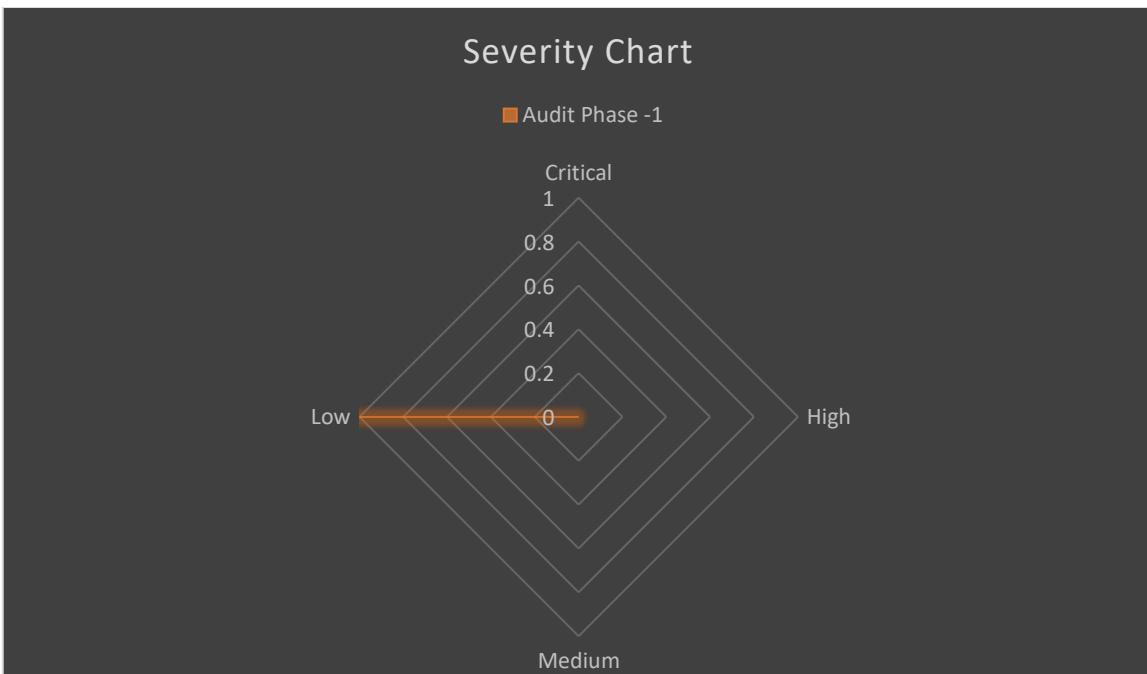
## Findings

### Summary

Here is a summary of our findings after scrutinizing the Nicho-NFT Smart Contract Review. During the first phase of our audit, we studied the smart contract source code and ran our in-house static code analyzer through the Specific tools. The purpose here is to statically identify known coding bugs, and then manually verify (reject or confirm) issues reported by tools. We further manually review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.

Severity	No. of Issues
Critical	0
High	0
Medium	0
Low	1
Total	1

# Smart Contract Audit



We have so far identified that there are potential issues with severity of **0 Critical, 0 High, 0 Medium, and 1 Low**. Overall, these smart contracts are well-designed and engineered.

# Smart Contract Audit

## Functional Overview

<code>(\\$)</code> = payable function	[Pub] public
<code>#</code> = non-constant function	[Ext] external
	[Prv] private
	[Int] internal

- + Context
  - [Int] \_msgSender
  - [Int] \_msgData
- + Ownable (Context)
  - [Pub] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner
  - [Int] \_transferOwnership #
- + [Int] IERC20
  - [Ext] totalSupply
  - [Ext] balanceOf
  - [Ext] transfer #

# Smart Contract Audit

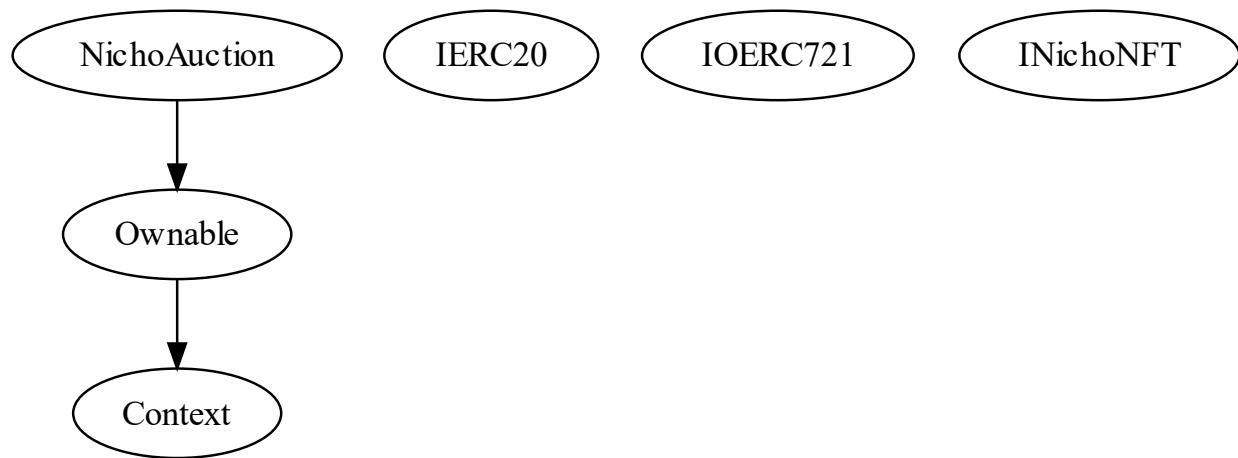
- [Ext] allowance
  - [Ext] approve #
  - [Ext] transferFrom #
- 
- + [Int] IOERC721
    - [Ext] tokenURI
    - [Ext] balanceOf
    - [Ext] ownerOf
    - [Ext] safeTransferFrom #
    - [Ext] approve #
    - [Ext] getApproved
- 
- + [Int] INichoNFT
    - [Ext] commissionFee
    - [Ext] denominator
    - [Ext] \_feeAddress
    - [Ext] whitelist
    - [Ext] blackList
- 
- + NichoAuction (Ownable)
    - [Pub] <Constructor> #
    - [Ext] <Fallback> (\$)
    - [Ext] addAuctionItem #
      - modifiers: tokenOwner,auctionNotExist

# Smart Contract Audit

- [Ext] cancelAuctionItem #
  - modifiers: auctionOwner,auctionExist
- [Ext] placeBid (\$)
  - modifiers: auctionExist,auctionStarted,auctionNotEnded
- [Ext] acceptBid #
  - modifiers: auctionOwner,auctionExist
- [Ext] cancelBid #
  - modifiers: auctionExist,auctionEnded
- [Ext] getBidsByBidder
- [Ext] isNFTListedToAuction
- [Ext] createOffer (\$)
  - modifiers: onlyTokenOwner
- [Ext] acceptOffer #
- [Ext] cancelOffer #
  - modifiers: onlyTokenOwner
- [Ext] withdrawETH #
  - modifiers: onlyOwner
- [Ext] withdrawTokens #
  - modifiers: onlyOwner

# Smart Contract Audit

## Inheritance



# Smart Contract Audit

## Detailed Results

### Issues Checking Status

#### 1. Floating Pragma

- SWC ID: 103
- Severity: Low
- Location: NichoAuction v2.sol
- Relationships: CWE-664: Improper Control of a Resource Through its Lifetime
- Description: A floating pragma is set. The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

```
7
8  pragma solidity ^0.8.0;
9
```

- Remediations: Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.

# Smart Contract Audit

## Automated Tool Results

Slither: -

```
NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503) sends eth to arbitrary user
  Dangerous calls:
    - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#494)
    - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#496)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

Reentrancy in NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503):
  External calls:
    - IERC721(tokenAddress).safeTransferFrom(msg.sender,item.creator,tokenId) (NichoAuction v2.sol#486)
  External calls sending eth:
    - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#494)
    - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#496)
  State variables written after the call(s):
    - item.isCancel = true (NichoAuction v2.sol#497)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities

NichoAuction.withdrawTokens(address,uint256) (NichoAuction v2.sol#524-528) ignores return value by IERC20(_token).transfer(msg.sender,_amount) (NichoAuction v2.sol#527)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

NichoAuction.auctionOwner(uint256) (NichoAuction v2.sol#265-270) uses a dangerous strict equality:
  - require(bool,string)(token.ownerOf(item tokenId) == msg.sender,Only Auction Owner) (NichoAuction v2.sol#268)
NichoAuction.getBidsByBidder(uint256) (NichoAuction v2.sol#439-454) uses a dangerous strict equality:
  - buyer.owner == msg.sender && buyer.bidPrice > 0 (NichoAuction v2.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

NichoAuction.getBidsByBidder(uint256).buyers (NichoAuction v2.sol#443) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

Reentrancy in NichoAuction.acceptBid(uint256,uint256) (NichoAuction v2.sol#386-414):
  External calls:
    - IERC721(item.tokenAddress).safeTransferFrom(msg.sender,buyer.owner,item.tokenId) (NichoAuction v2.sol#396)
  External calls sending eth:
    - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#404)
    - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#407)
  Event emitted after the call(s):
    - SoldOut(auctionId,buyer.bidPrice,buyer.owner) (NichoAuction v2.sol#408)
Reentrancy in NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503):
  External calls:
    - IERC721(tokenAddress).safeTransferFrom(msg.sender,item.creator,tokenId) (NichoAuction v2.sol#486)
  External calls sending eth:
    - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#494)
    - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#496)
  Event emitted after the call(s):
    - OffersSoldOut(tokenAddress,tokenId,msg.sender,item.creator) (NichoAuction v2.sol#502)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

NichoAuction.addAuctionItem(NichoAuction.ItemParam) (NichoAuction v2.sol#327-358) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(itemDeadline > block.timestamp,Invalid deadline) (NichoAuction v2.sol#334)
NichoAuction.getBidsByBidder(uint256) (NichoAuction v2.sol#439-454) uses timestamp for comparisons
  Dangerous comparisons:
    - i < item.buyers.length (NichoAuction v2.sol#445)
    - buyer.owner == msg.sender && buyer.bidPrice > 0 (NichoAuction v2.sol#447)
NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(item.expireTs >= block.timestamp,Offer already expired) (NichoAuction v2.sol#484)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503) compares to a boolean constant:
  - require(bool,string)(item.isCancel == false,Offer creator withdrew) (NichoAuction v2.sol#483)
NichoAuction.cancelOffer(address,uint256) (NichoAuction v2.sol#505-514) compares to a boolean constant:
  - require(bool,string)(item.isCancel == false,Already cancel) (NichoAuction v2.sol#509)
NichoAuction.notBlackList(address,uint256) (NichoAuction v2.sol#253-256) compares to a boolean constant:
  - require(bool,string)(NichoNFT.blacklist(tokenAddress,_tokenId) == false,tokenId is in blacklist) (NichoAuction v2.sol#254)
NichoAuction.auctionNotExist(address,uint256) (NichoAuction v2.sol#278-283) compares to a boolean constant:
  - require(bool,string)(item.exists != true,Auction Exists) (NichoAuction v2.sol#281)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Context._msgData() (NichoAuction v2.sol#25-27) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

# Smart Contract Audit

```
Pragma version^0.8.0 (NichoAuction v2.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function INichoNFT._feeAddress() (NichoAuction v2.sol#201) is not in mixedCase
Parameter NichoAuction.withdrawTokens(address,uint256)._token (NichoAuction v2.sol#524) is not in mixedCase
Parameter NichoAuction.withdrawTokens(address,uint256)._amount (NichoAuction v2.sol#524) is not in mixedCase
Variable NichoAuction.NichoNFT (NichoAuction v2.sol#207) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Reentrancy in NichoAuction.acceptBid(uint256,uint256) (NichoAuction v2.sol#386-414):
    External calls:
        - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#404)
        - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#407)
    Event emitted after the call(s):
        - SoldOut(auctionId,buyer.bidPrice,buyer.owner) (NichoAuction v2.sol#408)
Reentrancy in NichoAuction.acceptOffer(address,uint256,address) (NichoAuction v2.sol#480-503):
    External calls:
        - NichoNFT._feeAddress().transfer(_commissionValue) (NichoAuction v2.sol#494)
        - address(msg.sender).transfer(_sellerValue) (NichoAuction v2.sol#496)
    State variables written after the call(s):
        - item.isCancel = true (NichoAuction v2.sol#497)
    Event emitted after the call(s):
        - OfferSoldOut(tokenAddress,tokenId,msg.sender,item.creator) (NichoAuction v2.sol#502)
Reentrancy in NichoAuction.cancelBid(uint256) (NichoAuction v2.sol#417-436):
    External calls:
        - address(msg.sender).transfer(totalBidAmount) (NichoAuction v2.sol#434)
    Event emitted after the call(s):
        - CancelBid(auctionId,msg.sender) (NichoAuction v2.sol#435)
Reentrancy in NichoAuction.cancelOffer(address,uint256) (NichoAuction v2.sol#505-514):
    External calls:
        - address(msg.sender).transfer(item.price) (NichoAuction v2.sol#512)
    Event emitted after the call(s):
        - OfferCancel(tokenAddress,tokenId) (NichoAuction v2.sol#513)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4

renounceOwnership() should be declared external:
    - Ownable.renounceOwnership() (NichoAuction v2.sol#76-78)
transferOwnership(address) should be declared external:
    - Ownable.transferOwnership(address) (NichoAuction v2.sol#84-87)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

MythX: -

Report for NichoAuction v2.sol  
<https://dashboard.mythx.io/#/console/analyses/2adfba55-6280-4bc0-bf15-0db604be7180>

Line	SWC Title	Severity	Short Description
8	(SWC-103) Floating Pragma	Low	A floating pragma is set.
206	(SWC-123) Requirement Violation	Low	Requirement violation.
525	(SWC-123) Requirement Violation	Low	Requirement violation.

# Smart Contract Audit

Solhint: -

Linter results:

```
NichoAuction v2.sol:8:1: Error: Compiler version ^0.8.0 does not satisfy the requiresemver requirement
```

```
NichoAuction v2.sol:50:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

```
NichoAuction v2.sol:207:22: Error: Variable name must be in mixedCase
```

```
NichoAuction v2.sol:287:17: Error: Avoid to make time-based decisions in your business logic
```

```
NichoAuction v2.sol:293:17: Error: Avoid to make time-based decisions in your business logic
```

```
NichoAuction v2.sol:299:17: Error: Avoid to make time-based decisions in your business logic
```

```
NichoAuction v2.sol:321:5: Error: Explicitly mark visibility in function (Set ignoreConstructors to true if using solidity >=0.7.0)
```

# Smart Contract Audit

NichoAuction v2.sol:325:32: Error: Code contains empty blocks

NichoAuction v2.sol:330:9: Error: visibility modifier must be first in list of modifiers

NichoAuction v2.sol:334:36: Error: Avoid to make time-based decisions in your business logic

NichoAuction v2.sol:351:34: Error: Avoid to make time-based decisions in your business logic

NichoAuction v2.sol:363:9: Error: Visibility modifier must be first in list of modifiers

NichoAuction v2.sol:373:9: Error: Visibility modifier must be first in list of modifiers

NichoAuction v2.sol:389:9: Error: Visibility modifier must be first in list of modifiers

NichoAuction v2.sol:420:9: Error: Visibility modifier must be first in list of modifiers

NichoAuction v2.sol:466:9: Error: Visibility modifier must be first in list of modifiers

NichoAuction v2.sol:484:38: Error: Avoid to make time-based decisions in your business logic

NichoAuction v2.sol:507:9: Error: Visibility modifier must be first in list of modifiers

# Smart Contract Audit

## Basic Coding Bugs

No.	Name	Description	Severity	Result
1.	Constructor Mismatch	Whether the contract name and its constructor are not identical to each other.	Critical	PASSED
2.	Ownership Takeover	Whether the set owner function is not protected.	Critical	PASSED
3.	Redundant Fallback Function	Whether the contract has a redundant fallback function.	Critical	PASSED
4.	Overflows & Underflows	Whether the contract has general overflow or underflow vulnerabilities	Critical	PASSED
5.	Reentrancy	Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs	High	PASSED
6.	MONEY-Giving Bug	Whether the contract returns funds to an arbitrary address	High	PASSED
7.	Blackhole	Whether the contract locks ETH indefinitely: merely in without out	High	PASSED
8.	Unauthorized Self-Destruct	Whether the contract can be killed by any arbitrary address	Medium	PASSED
9.	Revert DoS	Whether the contract is vulnerable to DoS attack because	Medium	PASSED

# Smart Contract Audit

		of unexpected revert		
10.	Unchecked External Call	Whether the contract has any external call without checking the return value	Medium	PASSED
11.	Gasless Send	Whether the contract is vulnerable to gasless send	Medium	PASSED
12.	Send Instead of Transfer	Whether the contract uses send instead of transfer	Medium	PASSED
13.	Costly Loop	Whether the contract has any costly loop which may lead to Out-Of-Gas exception	Medium	PASSED
14.	(Unsafe) Use of Untrusted Libraries	Whether the contract use any suspicious libraries	Medium	PASSED
15.	(Unsafe) Use of Predictable Variables	Whether the contract contains any randomness variable, but its value can be predicated	Medium	PASSED
16.	Transaction Ordering Dependence	Whether the final state of the contract depends on the order of the transactions	Medium	PASSED
17.	Deprecated Uses	Whether the contract use the deprecated tx.origin to perform the authorization	Medium	PASSED
18.	Semantic Consistency Checks	Whether the semantic of the white paper is different from the implementation of the contract	Critical	PASSED

# Smart Contract Audit

## Conclusion

In this audit, we thoroughly analyzed Nicho-NFT's 'NichoAuction v2' Smart Contract. The current code base is well organized but there are promptly some low-level issues found in this phase of Smart Contract Audit.

Meanwhile, we need to call attention to that smart contract as a whole are still in an early, but exciting stage of development. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

# Smart Contract Audit

## About Virtual Caim

Just like our other parallel journey at eNebula Solution, we believe that people have a fundamental need to security and that the use of secure solutions enables every person to more freely use the Internet and every other connected technology. We aim to provide security consulting service to help others make their solutions more resistant to unauthorized access to data & inadvertent manipulation of the system. We support teams from the design phase through the production to launch and surely after.

The Virtual Caim is specifically incorporated to handle all kind of Security related operations, our Highly Qualified and Certified security team has skills for reviewing coding languages like Solidity, Rust, Go, Python, Haskell, C, C++ and JavaScript for common security vulnerabilities & specific attack vectors. The team has been reviewing implementations of cryptographic protocols and distributed system architecture, including in cryptocurrency, blockchains, payments, and smart contracts. Additionally, the team can utilize various tools to scan code & networks and build custom tools as necessary.

Although we are a small team, we surely believe that we can have a momentous impact on the world by being translucent & open about the work we do.

For more information about our other security services and consulting,  
please visit -- <https://virtualcaim.com/>  
& Mail us at – [audit@virtualcaim.com](mailto:audit@virtualcaim.com)