Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	WOM Protocol Pte. Ltd.
Website:	https://yaaas.me
Platform:	Ethereum
Language:	Solidity
Date:	June 28th, 2021

Table of contents

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Introduction

EtherAuthority was contracted by the Wom Protocol team to perform the Security audit of the Yaaas Exchange and Yaaas NFT (YAAAS) Token smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 25th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

YAAAS connects every kind of creator with hyped fans through the sale of NFT collectibles, exclusive collaborations and custom experiences. The YAAAS creator economy is brought by the WOM Protocol.

Audit scope

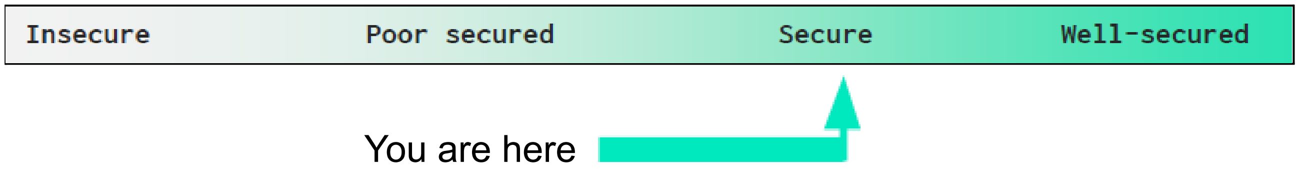
Name	**Code Review and Security Analysis Report for Woom Exchange and Yaaas NFT (YAAAS) Token Smart Contract**
Platform	**Ethereum / Solidity**
File 1	YaaasExchange.sol
Smart Contract Online Code	https://rinkeby.etherscan.io/address/0x24706abfe49689baa24dc3A3f96c791F12816a00#code
File 2	YaaasNFTtoken.sol
Smart Contract Online Code	https://rinkeby.etherscan.io/address/0xC0d477A730b519847f3D628657d8c899A1320AC8#code
Audit Date	June 28th, 2021
Revised Yaaas Exchange code	https://rinkeby.etherscan.io/address/0x15e905dea95aD21c44ECC4D0F5907c01D2079361#code
Revised Yaaas NFT Token contract	https://rinkeby.etherscan.io/address/0x34286d1f525a171a44a16a922b85d2c87da5393e#code

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: Yaaas Exchange - Owner Share: It is 1% by default and it can be changed by the owner. - SafeERC721 token only will work. - Users can set offer price, set for sale, set for auction, set expiry, etc.	**YES, This is valid.**
Name: Yaaas NFT (YAAAS) Token - ERC721 NFT token - Name: Yaaas NFT - Symbol: YAAAS - Anyone can mint new tokens without any fee (users pay only transaction gas cost)	**YES, This is valid.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contract is **Technically Secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We found 0 critical, 0 high, 0 medium and 1 low and some very low level technical issues.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Moderated
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	N/A
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: PASSED

Code Quality

This audit scope has 2 smart contracts. These smart contracts also contain Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the Wom Protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Wom Protocol .

The Wom Protocol team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given Wom Protocol smart contract code in the form of an Etherscan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://yaaas.me/ which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

(1) Interface

(a) IERC20

(b) IERC165

(c) IERC721

(d) IERC721Receiver

(2) Inherited contracts

(a) Context

(b) Ownable

(c) ERC721Validator

(3) Struct

(a) Offer

(b) Bid

(4) Usages

(a) using SafeMath for uint256;

(5) Events

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(b) event Swapped(address buyer, address seller, address token,uint256 assetId, uint256 price);

(c) event Listed(address seller,address collection,uint256 assetId,address token,uint256 price);

(d) event BidCreated(bytes32 id,address indexed collection,uint256 indexed assetId, address indexed bidder,address token,uint256 price,uint256 expiresAt);

(e) event BidSuccessful(address collection,uint256 assetId, address token,address bidder, uint256 price);

(f) event BidAccepted(bytes32 id);

(g) event BidCancelled(bytes32 id);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	addOffer	write	Passed	No Issue
2	setOfferPrice	write	Passed	No Issue
3	setForSell	write	Passed	No Issue
4	setForAuction	write	Passed	No Issue
5	setExpiresAt	write	Passed	No Issue
6	_getOwnerOffer	internal	Passed	No Issue
7	buyOffer	write	Passed	No Issue
8	_buyOffer	internal	Possible reentrency	Refer Audit Finding section
9	safePlaceBid	write	Passed	No Issue
10	setOwnerShare	modifier	Passed	No Issue
11	_createBid	internal	Passed	No Issue
12	cancelBid	write	Passed	No Issue
13	acceptBid	write	Passed	No Issue
14	onERC721Received	write	Passed	No Issue
15	onlyOwner	modifier	Passed	No Issue
16	transferOwnership	write	access only Owner	No Issue
17	renounceOwnership	write	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

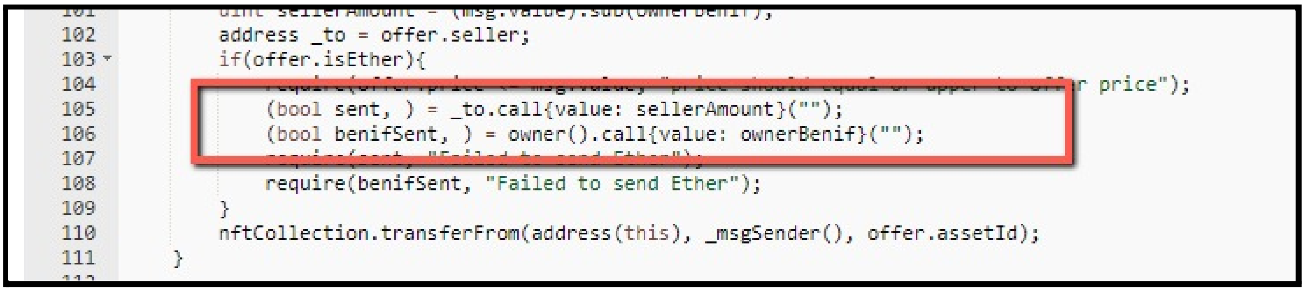
No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Low level function calls used in WoomExchange.sol



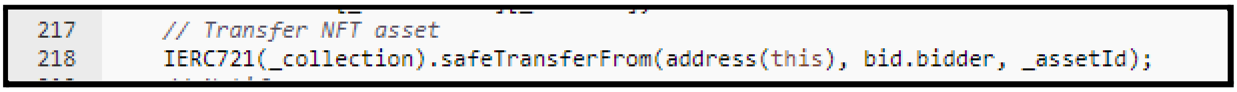
```
101            uint sellerAmount = (msg.value).sub(ownerBenif);
102            address _to = offer.seller;
103 ▾          if(offer.isEther){
104                require(offer.price <= msg.value, "price should equal or upper to offer price");
105                (bool sent, ) = _to.call{value: sellerAmount}("");
106                (bool benifSent, ) = owner().call{value: ownerBenif}("");
107                require(sent, "Failed to send Ether");
108                require(benifSent, "Failed to send Ether");
109            }
110            nftCollection.transferFrom(address(this), _msgSender(), offer.assetId);
111        }
```

Usually an attacker can create an attack smart contract and use it to reenter. In line number #105, if the _to is an attack smart contract, then sending ether to that wallet will trigger that attack contract's fallback function and either he can reenter or can call any other function which is dependent on _buyOffer function's state.

Resolution: To prevent such scenarios from rising, it is always better to use .transfer() to send the ether to any wallet as it only allocates 21,000 gas which is not enough to reenter.

Very Low / Discussion / Best practices:

(1) Use safeTransfer instead of safeTransferFrom in WoomExchange.sol



```
// Transfer NFT asset
IERC721(_collection).safeTransferFrom(address(this), bid.bidder, _assetId);
```

If the asset is being transferred from contract (address this) to user, then there is no need to use safeTranseferFrom. This will cause it to issue additional approval. Best practice is to just use the safeTransfer() method.

(2) All functions which are not called internally, must be declared as external. It is more efficient as sometimes it saves some gas. https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Centralization

This smart contract has some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- setOwnerShare: Owner can set his share.
- transferOwnership: Transfers ownership of the contract to a new account (`newOwner`).
- renounceOwnership: Renouncing ownership will leave the contract without an owner.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Technically Secured".**

The technical audit does not guarantee the ethical nature of the project and this audit report is never investment advice. All investors must do their due diligence before investing into the project.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

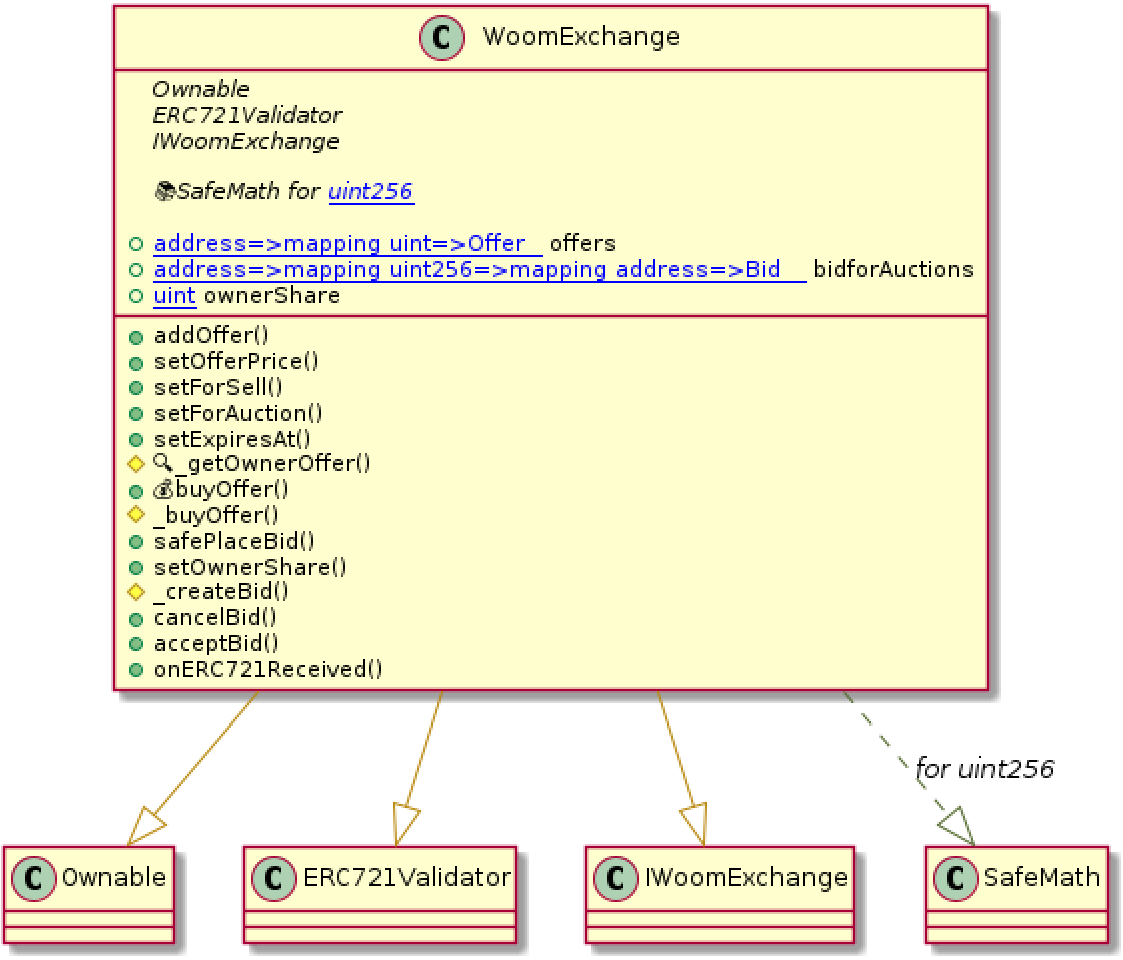
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

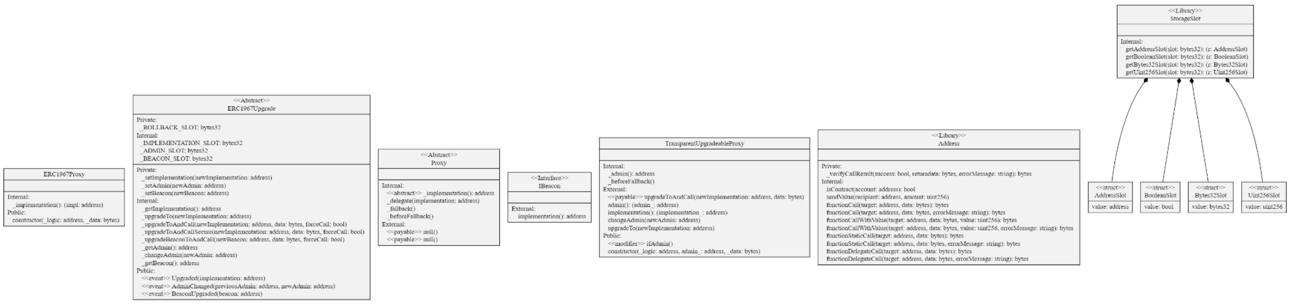
Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - Wom Protocol



Code Flow Diagram - Yaaas NFT (YAAAS)


ERC1967Proxy
Internal:
_implementation(): (impl: address)
Public:
constructor(_logic: address, _data: bytes)
<<Abstract>>
ERC1967Upgrade
Private:
_ROLLBACK_SLOT: bytes32
Internal:
_IMPLEMENTATION_SLOT: bytes32
_ADMIN_SLOT: bytes32
_BEACON_SLOT: bytes32
Private:
_setImplementation(newImplementation: address)
_setAdmin(newAdmin: address)
_setBeacon(newBeacon: address)
Internal:
_getImplementation(): address
_upgradeTo(newImplementation: address)
_upgradeToAndCall(newImplementation: address, data: bytes, forceCall: bool)
_upgradeToAndCallSecure(newImplementation: address, data: bytes, forceCall: bool)
_upgradeBeaconToAndCall(newBeacon: address, data: bytes, forceCall: bool)
_getAdmin(): address
_changeAdmin(newAdmin: address)
_getBeacon(): address
Public:
<<event>> Upgraded(implementation: address)
<<event>> AdminChanged(previousAdmin: address, newAdmin: address)
<<event>> BeaconUpgraded(beacon: address)
<<Abstract>>
Proxy
Internal:
<<abstract>> _implementation(): address
_delegate(implementation: address)
_fallback()
_beforeFallback()
External:
<<payable>> null()
<<payable>> null()
<<Interface>>
IBeacon
External:
implementation(): address
TransparentUpgradeableProxy
Internal:
_admin(): address
_beforeFallback()
External:
<<payable>> upgradeToAndCall(newImplementation: address, data: bytes)
admin(): (admin_: address)
implementation(): (implementation_: address)
changeAdmin(newAdmin: address)
upgradeTo(newImplementation: address)
Public:
<<modifier>> ifAdmin()
constructor(_logic: address, admin_: address, _data: bytes)
<<Library>>
Address
Private:
_verifyCallResult(success: bool, returndata: bytes, errorMessage: string): bytes
Internal:
isContract(account: address): bool
sendValue(recipient: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: string): bytes
functionStaticCall(target: address, data: bytes): bytes
functionStaticCall(target: address, data: bytes, errorMessage: string): bytes
functionDelegateCall(target: address, data: bytes): bytes
functionDelegateCall(target: address, data: bytes, errorMessage: string): bytes
<<Library>>
StorageSlot
Internal:
getAddressSlot(slot: bytes32): (r: AddressSlot)
getBooleanSlot(slot: bytes32): (r: BooleanSlot)
getBytes32Slot(slot: bytes32): (r: Bytes32Slot)
getUint256Slot(slot: bytes32): (r: Uint256Slot)
<<struct>>
AddressSlot
value: address
<<struct>>
BooleanSlot
value: bool
<<struct>>
Bytes32Slot
value: bytes32
<<struct>>
Uint256Slot
value: uint256

Slither Results Log

```
INFO:Detectors:
WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221) ignores return
value by IERC20(bid.token).transferFrom(bid.bidder,_msgSender(),sellerAmount)
(WoomExchange.sol#213)
WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221) ignores return
value by IERC20(bid.token).transferFrom(bid.bidder,owner(),ownerBenif)
(WoomExchange.sol#214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
WoomExchange._buyOffer(IWoomExchange.Offer,address) (WoomExchange.sol#98-111)
performs a multiplication on the result of a division:
-ownerBenif = (msg.value).div(100).mul(ownerShare) (WoomExchange.sol#100)
WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221) performs a
multiplication on the result of a division:
-ownerBenif = (bid.price).div(100).mul(ownerShare) (WoomExchange.sol#201)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221):
External calls:
- IERC20(bid.token).transferFrom(bid.bidder,_msgSender(),sellerAmount)
(WoomExchange.sol#213)
- IERC20(bid.token).transferFrom(bid.bidder,owner(),ownerBenif)
(WoomExchange.sol#214)
State variables written after the call(s):
- delete offers[_collection][_assetId] (WoomExchange.sol#216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancyvulnerabilities-
1
INFO:Detectors:
Reentrancy in WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221):
External calls:
- IERC20(bid.token).transferFrom(bid.bidder,_msgSender(),sellerAmount)
(WoomExchange.sol#213)
- IERC20(bid.token).transferFrom(bid.bidder,owner(),ownerBenif)
(WoomExchange.sol#214)
- IERC721(_collection).safeTransferFrom(address(this),bid.bidder,_assetId)
(WoomExchange.sol#218)
Event emitted after the call(s):
- BidSuccessful(_collection,_assetId,bid.token,bid.bidder,bid.price)
(WoomExchange.sol#220)
Reentrancy in
WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)
(WoomExchange.sol#15-58):
External calls:
- nftCollection.safeTransferFrom(_seller,address(this),_assetId) (WoomExchange.sol#55)
Event emitted after the call(s):
- Listed(_seller,_collection,_assetId,token,_price) (WoomExchange.sol#56)
Reentrancy in WoomExchange.buyOffer(address,uint256) (WoomExchange.sol#90-97):
External calls:
- _buyOffer(offer,collection) (WoomExchange.sol#94)
- (sent) = _to.call{value: sellerAmount}() (WoomExchange.sol#105)
- (benifSent) = owner().call{value: ownerBenif}() (WoomExchange.sol#106)
- nftCollection.transferFrom(address(this),_msgSender(),offer.assetId)
(WoomExchange.sol#110)
External calls sending eth:
- _buyOffer(offer,collection) (WoomExchange.sol#94)
- (sent) = _to.call{value: sellerAmount}() (WoomExchange.sol#105)
- (benifSent) = owner().call{value: ownerBenif}() (WoomExchange.sol#106)
```

```
Event emitted after the call(s):
- Swapped(_msgSender(),offer.seller,collection,assetId,msg.value) (WoomExchange.sol#95)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancyvulnerabilities-
3
INFO:Detectors:
WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221) uses
timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(bid.expiresAt <= block.timestamp,Marketplace: the bid expired)
(WoomExchange.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Different versions of Solidity is used:
- Version used: ['>=0.4.22<0.9.0', '^0.8.0']
- ^0.8.0 (Context.sol#3)
- >=0.4.22<0.9.0 (ERC721Validator.sol#1)
- ^0.8.0 (IERC165.sol#3)
- ^0.8.0 (IERC20.sol#3)
- ^0.8.0 (IERC721.sol#3)
- ^0.8.0 (IERC721Receiver.sol#3)
- >=0.4.22<0.9.0 (IWoomExchange.sol#1)
- ^0.8.0 (Ownable.sol#3)
- >=0.4.22<0.9.0 (SafeMath.sol#1)
- >=0.4.22<0.9.0 (WoomExchange.sol#1)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragmadirectives-
are-used
INFO:Detectors:
Context._msgData() (Context.sol#20-23) is never used and should be removed
SafeMath.add(uint256,uint256) (SafeMath.sol#25-29) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (Context.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (ERC721Validator.sol#1) is too complex
Pragma version^0.8.0 (IERC165.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (IERC20.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (IERC721.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (IERC721Receiver.sol#3) necessitates a version too recent to be trusted.
Consider deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (IWoomExchange.sol#1) is too complex
Pragma version^0.8.0 (Ownable.sol#3) necessitates a version too recent to be trusted. Consider
deploying with 0.6.12/0.7.6
Pragma version>=0.4.22<0.9.0 (SafeMath.sol#1) is too complex
Pragma version>=0.4.22<0.9.0 (WoomExchange.sol#1) is too complex
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-ofsolidity
INFO:Detectors:
Low level call in WoomExchange._buyOffer(IWoomExchange.Offer,address)
(WoomExchange.sol#98-111):
- (sent) = _to.call{value: sellerAmount}() (WoomExchange.sol#105)
- (benifSent) = owner().call{value: ownerBenif}() (WoomExchange.sol#106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
WoomExchange (WoomExchange.sol#9-226) should inherit from IERC721Receiver
(IERC721Receiver.sol#10-21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-inheritance
INFO:Detectors:
Parameter
WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)._seller
(WoomExchange.sol#16) is not in mixedCase
Parameter
WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)._collect
```

```
ion (WoomExchange.sol#17) is not in mixedCase
Parameter
WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)._assetId
(WoomExchange.sol#18) is not in mixedCase
Parameter
WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)._price
(WoomExchange.sol#21) is not in mixedCase
Parameter WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)._collection
(WoomExchange.sol#114) is not in mixedCase
Parameter WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)._assetId
(WoomExchange.sol#115) is not in mixedCase
Parameter WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)._token
(WoomExchange.sol#116) is not in mixedCase
Parameter WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)._price
(WoomExchange.sol#117) is not in mixedCase
Parameter WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)._expiresAt
(WoomExchange.sol#118) is not in mixedCase
Parameter WoomExchange.cancelBid(address,uint256,address)._collection
(WoomExchange.sol#178) is not in mixedCase
Parameter WoomExchange.cancelBid(address,uint256,address)._assetId (WoomExchange.sol#179)
is not in mixedCase
Parameter WoomExchange.cancelBid(address,uint256,address)._bidder (WoomExchange.sol#180)
is not in mixedCase
Parameter WoomExchange.acceptBid(address,uint256,address)._collection
(WoomExchange.sol#191) is not in mixedCase
Parameter WoomExchange.acceptBid(address,uint256,address)._assetId (WoomExchange.sol#192)
is not in mixedCase
Parameter WoomExchange.acceptBid(address,uint256,address)._bidder (WoomExchange.sol#193)
is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-soliditynaming-
conventions
INFO:Detectors:
Redundant expression "this (Context.sol#21)" inContext (Context.sol#15-24)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
ERC721Validator._INTERFACE_ID_ERC721 (ERC721Validator.sol#4) is never used in
WoomExchange (WoomExchange.sol#9-226)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
- Ownable.renounceOwnership() (Ownable.sol#54-57)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (Ownable.sol#63-67)
addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256) should be declared
external:
- WoomExchange.addOffer(address,address,uint256,address,bool,uint256,bool,bool,uint256)
(WoomExchange.sol#15-58)
setOfferPrice(address,uint256,uint256) should be declared external:
- WoomExchange.setOfferPrice(address,uint256,uint256) (WoomExchange.sol#59-63)
setForSell(address,uint256,bool) should be declared external:
- WoomExchange.setForSell(address,uint256,bool) (WoomExchange.sol#64-74)
setForAuction(address,uint256,bool) should be declared external:
- WoomExchange.setForAuction(address,uint256,bool) (WoomExchange.sol#75-79)
setExpiresAt(address,uint256,uint256) should be declared external:
- WoomExchange.setExpiresAt(address,uint256,uint256) (WoomExchange.sol#80-84)
buyOffer(address,uint256) should be declared external:
- WoomExchange.buyOffer(address,uint256) (WoomExchange.sol#90-97)
safePlaceBid(address,uint256,address,uint256,uint256) should be declared external:
- WoomExchange.safePlaceBid(address,uint256,address,uint256,uint256)
(WoomExchange.sol#113-121)
setOwnerShare(uint256) should be declared external:
- WoomExchange.setOwnerShare(uint256) (WoomExchange.sol#122-127)
acceptBid(address,uint256,address) should be declared external:
- WoomExchange.acceptBid(address,uint256,address) (WoomExchange.sol#190-221)
onERC721Received(address,address,uint256,bytes) should be declared external:
```

EtherAuthority

Email: audit@EtherAuthority.io