SECURICHAIN

SMART CONTRACT AUDIT

SECURITY ANALYSIS REPORT FOR

MONSTERRA

May 27th , 2022

SECURICHAIN

Security Rating

The rating is based on the number, severity and latest status of detected issues

Monsterra

SECURICHAIN

DISCLAIMER

This report containing confidential information which can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

SecuriChain does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed.

The report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

Monsterra

SECURICHAIN

TABLE OF CONTENTS

Monsterra

SECURICHAIN

VULNERABILITY ASSESSMENT OVERVIEW

1.1. ASSIGNING RISK LEVELS

The Auditor categorizes each of the detected vulnerabilities into 4 levels (High, Medium, Low, and Info) according to the degree of the risks it may cause in the Customer's operations. For details of the rating standards, please refer to "Appendix 2 Risk Rating." Please also note that the assessment of the findings is based on Auditor's own perspective and may contain speculations in some cases.

SECURICHAIN

1.2. SCOPE OF WORK

Project Name	**Monsterra Contract**
Platform	**Ethereum**
Languages	**Solidity**
Methods	**Automation scan, architecture review, functional testing, manual code review**
Repository	**https://github.com/sota-finance/game-market-contracts/tree/develop** **commit : af89162b1eb087baab807dc7ed363cd70a6cc90d**
Documents	**None**
Timelines	**May 14th, 2022 – May 27th, 2022**

SECURICHAIN

1.3. CHECKSUM FILE

GAME-MARKET-CONTRACTS-DEVELOP

No.	Hash	Name
1	31bee091ae64929164a795e0c5c68366ba7c4f48df41fd77ad642c9eddce9187	GameERC721.sol
2	8ff6500a967cb817f9a854816dcd5808e457fc9219364e90fbcad228d957180c	GameMarketPayment.sol
3	c675cd4cbeac95f3e3ab6a8f10f91534ea72f1f2ef5b59df9141c82d69aa2526	GameMarket.sol
4	1147f558792d966f9f5a1b8b333d4e5fa521b1f4255cfea3abf51a828f274403	Manager.sol
5	368782941c80e6d245299719adc94c85cd7823e1cd621aeff17671d8a64a94bb	MiniGameContract.sol
6	a3618b24f0fbad68448991be52ce4d413ca58fb5b2c95cdfa8313af24d12d6a9	MockERC721.sol
7	b5ad859da71ec400c911f861338a3554cb1f817ce7bd372c4cdc35da37b6561d	MockMON.sol
8	1d50b918a19ddfc64e4de90fb31abd1371d195386a06c8e8a4d4cff38685a80d	MonsConvertContract.sol

SECURICHAIN

No.	Hash	Name
9	89d7179873e677d23abbb2b1782d2e9ad512c719bcf96ee1fa9bc66658e71d8c	MonsterraLandNFT.sol
10	89d7179873e677d23abbb2b1782d2e9ad512c719bcf96ee1fa9bc66658e71d8c	MonsterraMonsterNFT.sol
11	64a0680c3da02da48039905a7016d51d7ce21e9ae81542e1f0988fe2b24ddd6d	MonsterraSkillNFT.sol
12	5e655d0f0acc46f4c73e4cbe82294c6d7c7eab166cd29ef38d53587575e36a4c	MonsterraSoulCoreNFT.sol
13	a80d84230b96993b97a63484f26f9ea06c635ba95ae9349e25344247a537552e	MysteryBoxManager.sol
14	e09131bcd869b90f42302323fb86f2132efa43487d8ddcf49149784985bee802	MysteryBoxNFT.sol
15	8926e347994db8dc0d34ce647b2fa224ab16cb2adb5d0064bb355889965c17e2	Token.sol

SECURICHAIN

1.4. ASSESSMENT RESULTS

According to the assessment, the Customer's smart contracts have a security rating of 96/100

RATE	DESCRIPTION
96-100	**No vulnerabilities** were found or all detected ones have been resolved
70-95	Unresolved **Low-level** vulnerabilities exist
40-69	Unresolved **Medium-level** vulnerabilities exist
0-39	Unresolved **High-level** vulnerabilities exist

For more information on criteria for risk rating, refer to Appendix.2

SECURICHAIN

FINDINGS

2.1 List of Vulnerabilities

The detected vulnerabilities are listed below. Please refer to "Appendix.2 Risk Rating" for the risk assessment method.

Vulnerabilities distributed in the smart contract

ID	Risk Level	Name	Amount	Status
SC1	High	Anyone can mint Tokens	5	Resolved
SC2	High	Anyone can mint NFT	1	Resolved
SC3	High	Reentrancy Vulnerabilities	7	Resolved
SC4	High	Business Logic Vulnerability	1	Resolved
SC5	Medium	Check the wrong lending conditions	1	Resolved
SC6	Medium	Anyone can cancel Landing	1	Resolved
SC7	Medium	Misbehaving Function	1	Resolved
SC8	Medium	Misbehaving Function	1	Resolved
SC9	Medium	Missing Zero Address Validation	10	Resolved

SECURICHAIN

ID	Risk Level	Name	Amount	Status
SC10	Medium	Uninitialized State Variables	6	Acknowledged
SC11	Medium	Bad Randomness	1	Resolved
SC12	Low	Uninitialized State Variables	1	Resolved
SC13	Low	Misbehaving Function	1	Resolved
SC14	Low	Unused State Variable	7	Acknowledged
SC15	Low	Public function that could be declared external	20	Acknowledged
SC16	Low	Unnecessary override	1	Resolved
SC17	Information	Unlocked Pragma	1	Resolved

For rating each vulnerability, refer to Appendix 2.

SECURICHAIN

2.1 Details

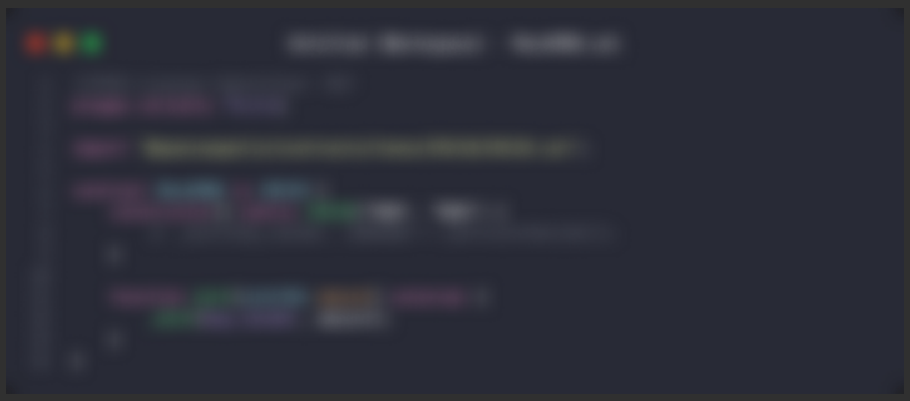
[1] Anyone can mint Tokens

High: 1

Overview

Of all Monstera's mint() token functions, the mint() function is external, so anyone can mint as many tokens as they want..
It can be exploited by bad guys, creating too many tokens, and at the same time leading to the loss of token value

Example: MockMON.mint()#L11-13



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Location

MockMON.mint()#L11-13

MockERC721.mint()#L10-12

Token.mintToken()#L11-L13

ItemsTest.mintToken()#l11-L13

ItemsTest2.mintToken()#l11-L13

PoC:https://testnet.bscscan.com/token/0x8e62425c4f951ad95c0f10408f585bba3bd0d890

Recommendation

Add the onlyOwner modifier for owner-only functions.

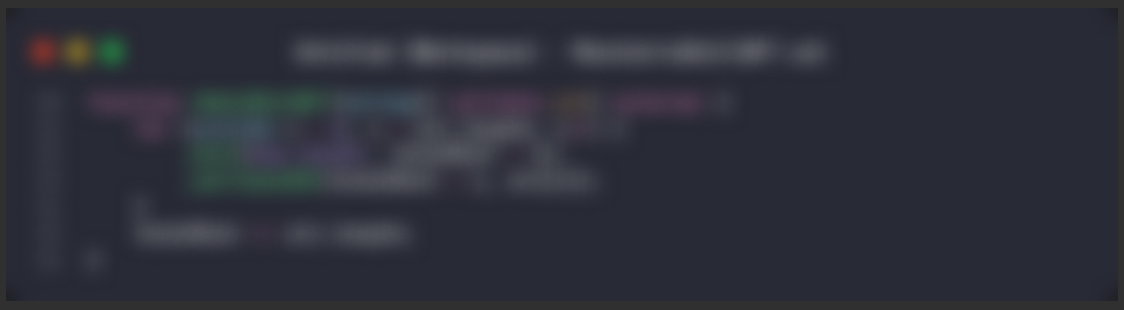
SECURICHAIN

[2] Anyone can mint NFT

High: 1

Overview

Missing onlyOwner check in AdminMintNFT() function, so anyone can call it.
It can be exploited by bad guys, creating too many tokens, and at the same time leading to the loss of token value.



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Location

MonsterraSkillNFT.AdminMintNFT()#L20

Recommendation

Add the onlyOwner modifier for owner-only functions.

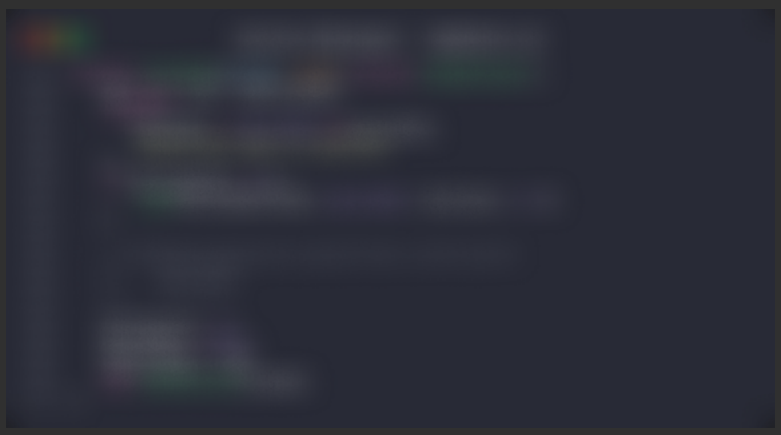
SECURICHAIN

[3] Reentrancy Vulnerabilities

High: 7

Overview

Paying before updating the value of variables related to the remaining amount can lead to the risk of re-entrancy attack.
The attacker can withdraw all the money in the contract.
Example: GameMarket.cancelBid()#L624-#642



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

The _paid() function paid the token to the caller and then updated the bid.quatity = 0.

Location

GameMarket.cancelBid()#L624-#642
GameMarket.buy()#L292-315
GameMarket.UpdateBid()#L644-704
GameMarket.acceptBid()#L706-779
GameMarket.cancelOrder()#L394-420
GameMarket.updateOrder()#L347-392
GameMarket.buyBundle()#L488-538

PoC:
https://testnet.bscscan.com/tx/0xaa67f71c495bd96b9a26434a17d7f8dbdbe2f61b7b708e0cb18099245e091642

Recommendation

You can use ReentrancyGuardUpgradeable(@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol). Using nonReentrant with token sending functions.

SECURICHAIN

[4] Business Logic Vulnerability

High: 1

Overview

The lendNFT() function transfers the token to the caller (rent) but has no strings attached, other than that the tenant only pays the rental fee (lendNft.fee).

Tenants can own the product at a low cost (lendingNft.fee)

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Location

GameERC721.lendNFT()#L87

Recommendation

Check the logic of lending.

SECURICHAIN

[5] Bad Randomness

High: 1

Overview

The genrand_int32() function generates an unsafe seed, the use of a block.difficulty, block.timestamp is predictable.

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Recommendation

Using Chainlink VRF

Location

MysteryBoxManager.genrand_int32()#L79-80

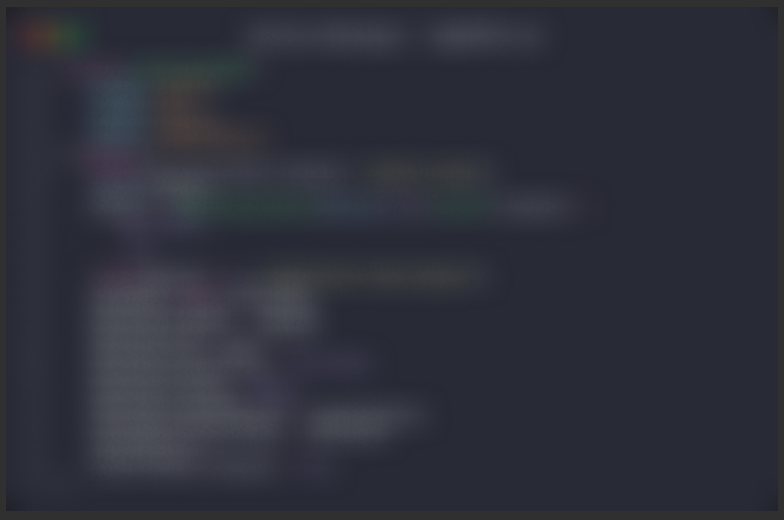
SECURICHAIN

[6] Check the wrong lending conditions

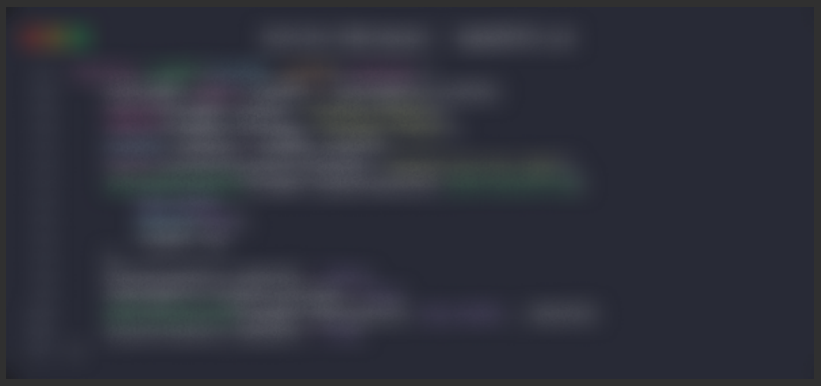
Medium: 1

Overview

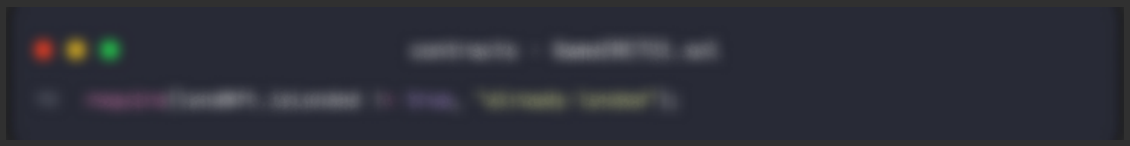
Check for incorrect rental condition. The variable lendNft.isLended is initialized to false, so require(lendNft.isLended, "already-lended") always returns false.



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)



Recommendation



Location

GameERC721.lendNFT#L90

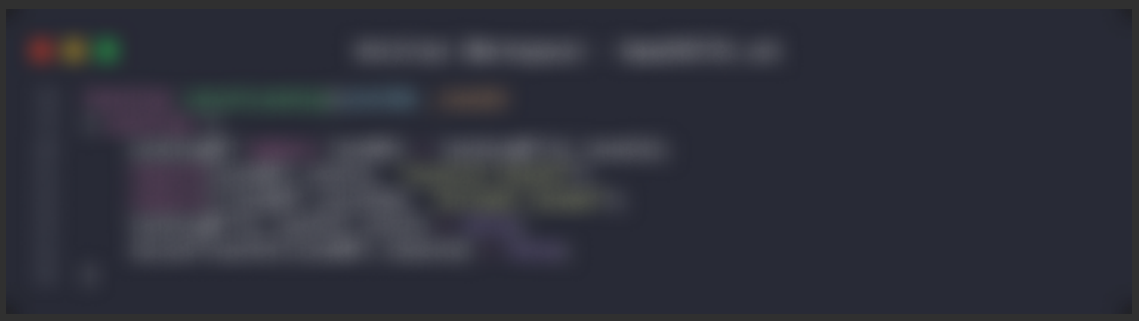
SECURICHAIN

[7] Anyone Call Cancel Landing

Medium: 1

Overview

The cancelLanding() function is external and does not impose any restrictions on the caller, so anyone can cancel any _lendId



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Recommendation

Add owner to check step.

Location

GameERC721.cancelLanding()#L78

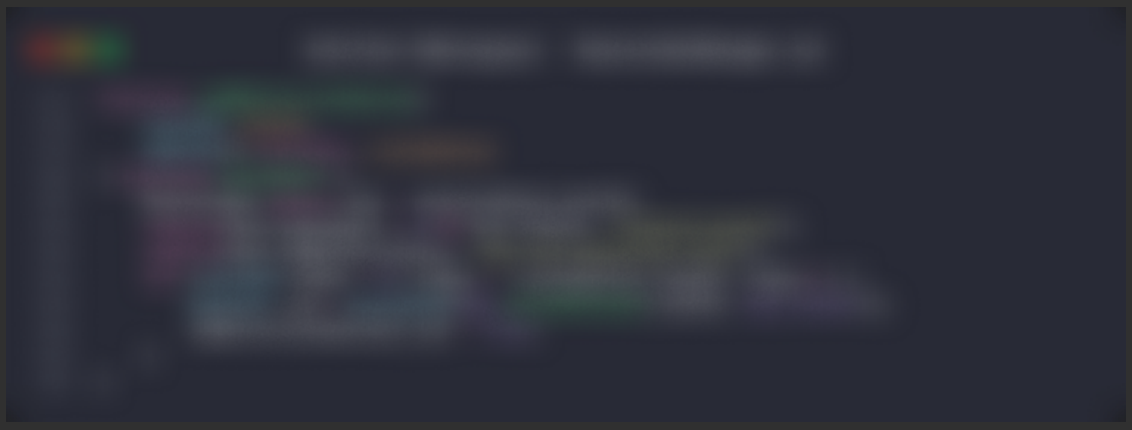
SECURICHAIN

[8] Misbehaving Function

Medium: 1

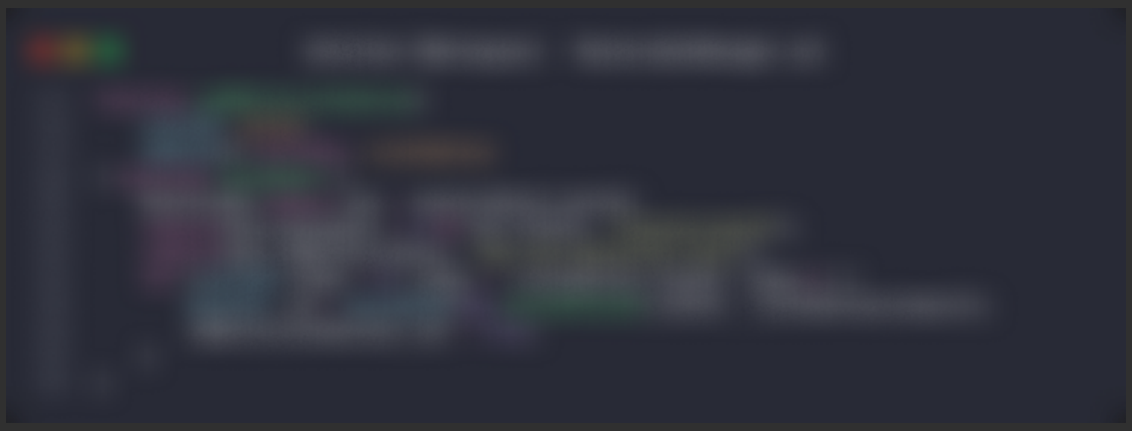
Overview

The address is added to the Whitelist using the addWhiteListAddress() function, but the added address is **msg.sender**



Recommendation

The code can be modified as follows:



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Location

MysteryBoxManager.addWhiteListAddress()#L185

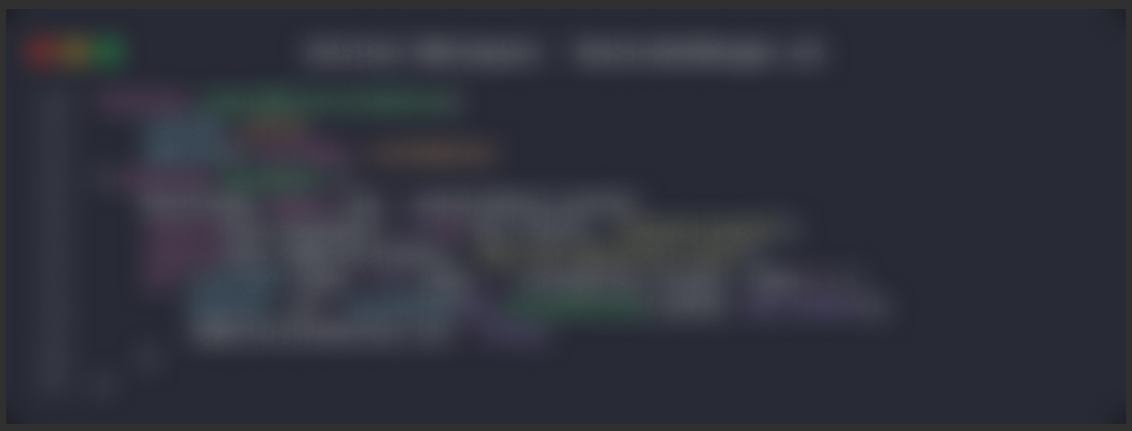
SECURICHAIN

[9] Misbehaving Function

Medium: 1

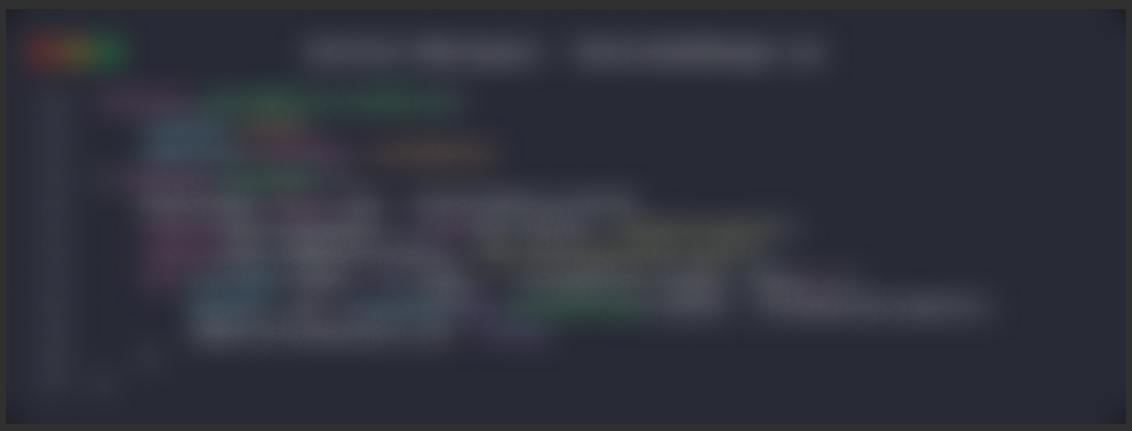
Overview

The removeWhiteListAddress() function is used to remove the addresses in the Whitelist, but the removed address is msg.sender



Recommendation

The code can be modified as follows:



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Location

MysteryBoxManager.removeWhiteListAddress()#L198

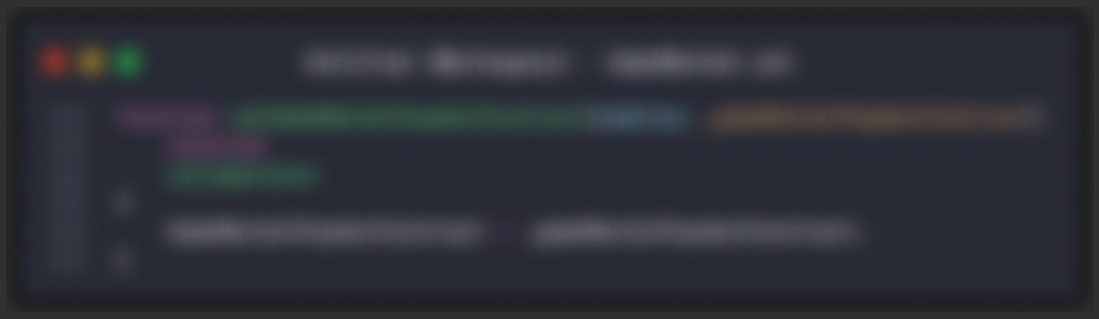
SECURICHAIN

[10] Missing Zero Address Validation

Medium: 1

Overview

Calling a function without parameters can result in Solidity assigning a value itself (ex: address is set to 0). Lack of checking for non-zero addresses can lead to asset loss when performing transactions or setting payment addresses, etc.
Example:
GameMarket.setGameMarketPaymentContract()._gameMarketPaymentContract#L426



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

If you accidentally call the setGameMarketPaymentContract() function without passing any parameters, the payment address may be set to 0

Recommendation

Check that the address is not zero.

Location

GameMarket.setGameMarketPaymentContract()._gameMarketPaymentContract#L426
GameMarket.withdrawFunds()._beneficiary#L892-911

SECURICHAIN

MiniGameContract.withdrawFunds()._beneficiary()L255-275
MonsterraConvertContract.initialize()#L57 (all parameter)
MonsterraConvertContract.updateSignAddress()._address#L84
MonsterraConvertContract.updateBinanceNFTsContract()._address#L88
MonsterraConvertContract.updateMysteryBoxContract()._address#L92
MysteryBoxManager.setMysteryBoxContract()._contractAddress#L220
MysteryBoxManager.withdrawFunds()._beneficiary#L236
GameERC721.lendNFT().createLendingNFT#L90

Monsterra

SECURICHAIN

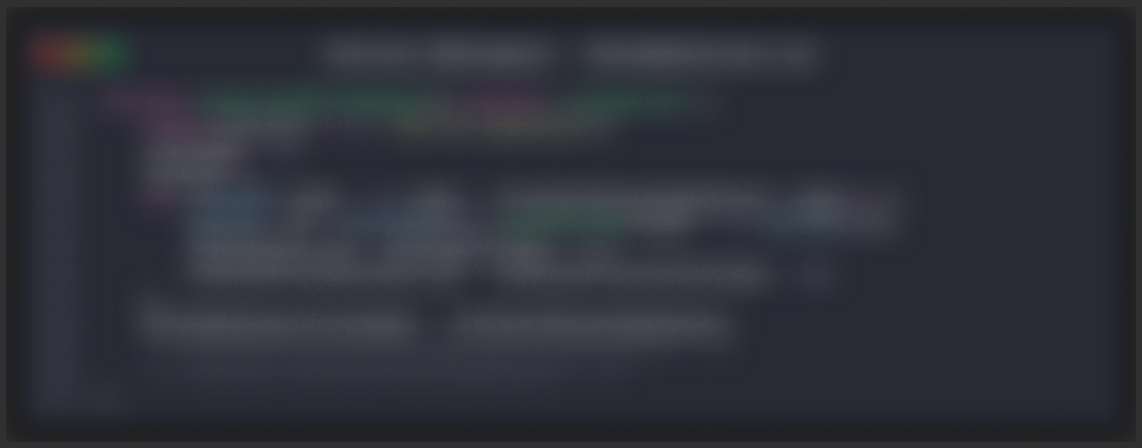
[11] Uninitialized State Variables

Medium: 6

Overview

Variables are initialized but not assigned a value, which leads to incorrect contract logic

Example: MiniGameConTract.threeCardPockerGameSeason#L21



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

threeCardPockerGameSeason is declared but not initialized. As a result, migrationMiniGameData()#l123-127 will never work and MiniGameSeason[totalGame] will always be 0.

Recommendation

Reconsider variable initialization

Location

MiniGameConTract.threeCardPockerGame#L20
MiniGameConTract.threeCardPockerGameSeason#L21
MiniGameConTract.totalCoefficientFee#L24
MiniGameConTract.totalUser#L25
MysteryBoxManager.listItemsCalculate#L26
GameMarket._acceptBid().order.isOnsale#L728

SECURICHAIN

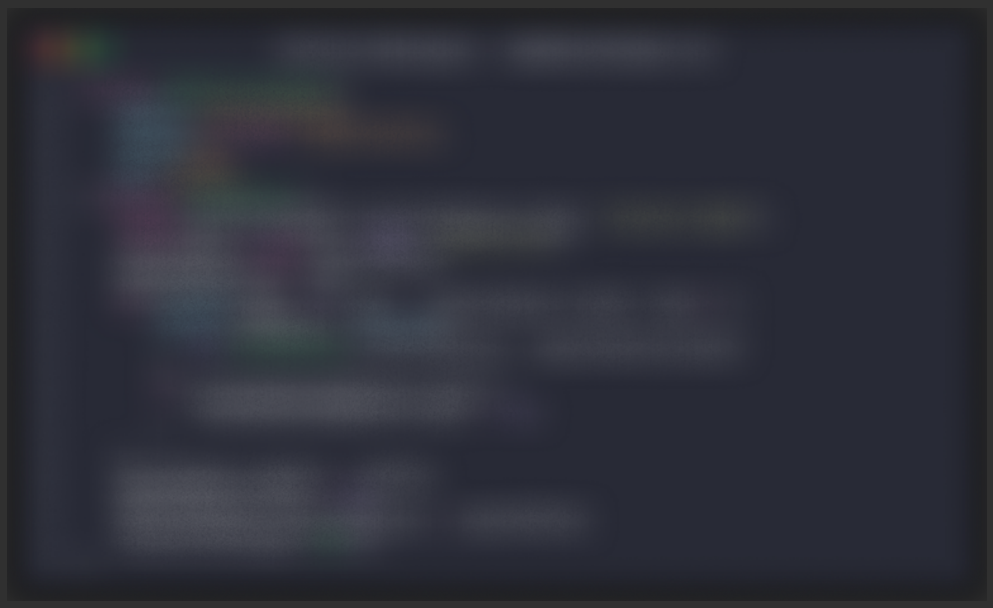
[12] Uninitialized State Variables

Low: 1

Overview

After each time addContractSupport() is called, the totalContractSupport variable is incremented by 1.

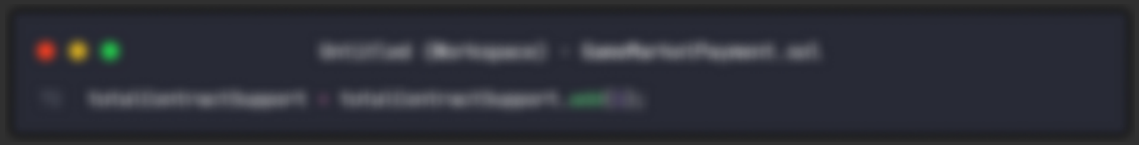
However, the current treatment: totalContractSupport.add(1) as the totalContractSupport variable is not incremented by 1 as intended.



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Recommendation

The code can be modified as follows:



Location

GameMarketPayment.addContractSupport()#L72

SECURICHAIN

[13] Misbehaving Function

Low: 1

Overview

Setting the wrong ItemMapBundle value leads to the case that NFTs purchased from the bundle will fail at the acceptBid().

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

When buying a bundle, the value at ItemMapBundle is set to 0 by calculating the hash from the arguments:

bundle.listTokenAddress[index]
bundle.listTokenId[index]
msg.sender

In the 3rd argument, instead of getting the bundle owner's address, the program takes the **bundle** buyer's address (msg.sender) leading to the wrong value.

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

As a result, the line GameMarket.acceptBid()#L730 returns a false result.

Recommendation

Change the msg.sender at the line GameMarket.buyBundle()#L519 to the address of the bundle owner.

Location

GameMarket.buyBundle()#L519

SECURICHAIN

[14] Unused State Variable

Low: 7

Overview

There is a fee to store and change data on the blockchain. Declared but unused variables waste gas.
Example: GameMarket._withdrawAmount#L900

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

The variable _withdrawAmount at line #L900 is not used for any purposes.

Recommendation

Delete unnecessary variables if they're not in use.

Location

GameMarket._version#L72 (declared but not used)
GameMarket._withdrawAmount#L900
GameMarketPayment.maxUint#L14
MiniGameContract._withdrawAmount#L263
MonsConvertContract.randomId#L27
MysteryBoxNFT.totalBuy#L21
MysteryBoxManager._withdrawAmount#L239

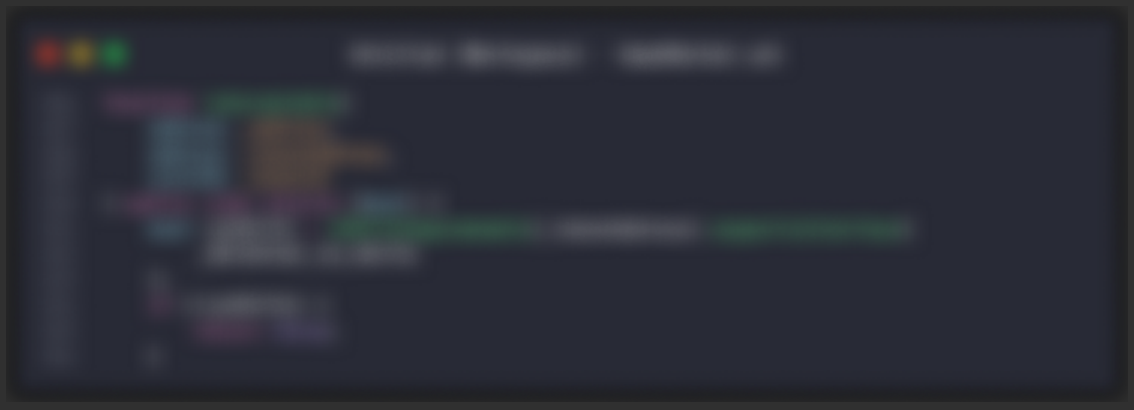
SECURICHAIN

[15] Public function that could be declared external

Low: 20

Overview

Public functions that are never called by the contract should be declared external to save gas.
Example: GameMarket.isAcceptable()#L856



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Recommendation

Use the external attribute for functions never called from the contract.

Location

GameMarket.initialize()#L125
GameMarketPayment.initialize()#L34
GameMarket.getBundleInfo()#L835
GameMarket.isAcceptable()#L856
Manager.pause()#L9
Manager.unPause()#L13
MonsConvertContract.initialize()#L57

SECURICHAIN

MonsterraMonsterNFT.initialize()#L14
MonsterraSoulCoreNFT.initialize()#L14
MonsterraLandNFT.initialize()#L14
MonsterraSkillNFT.initialize()#L14
MiniGameContract.initialize()#L55
MiniGameContract.isAddressJoinedSeasonTCP()#L156
MiniGameContract.remainingTurnCanBuy()#236
MysteryBoxManager.initialize()#64
MysteryBoxManager.genrand_int32()#70
MysteryBoxManager.getTotalBuy()#203
MysteryBoxManager.getCalculate()#212
MysteryBoxManager.isCanBuyBox()#223
MysteryBoxNFT.initialize()#L24

SECURICHAIN

[16] Unnecessary override

Low: 3

Overview

It is unnecessary and wasteful to override some transfer functions just to check for isLockTransfer [tokenId].

(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Recommendation

Write a modifier function to check `require(!isLockTransfer[tokenId], "tokenId-locked")`.
For functions that need to check `isLockTransfer`, add this modifier

Location

GameERC721.safeTransferFrom()#L104-115
GameERC721.safeTransferFrom()#L117-130
GameERC721.transferFrom()#L132-144

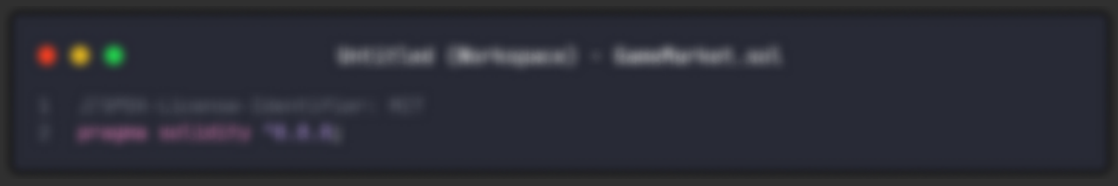
SECURICHAIN

[17] Unlocked Pragma

Info: 1

Overview

Contracts should be deployed with the same compiler version and flags that they have been thoroughly tested. Locking the pragma helps to ensure that contracts do not accidentally get deployed using.



(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

An outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the chosen compiler version.
Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case of contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma to compile it locally.

Location

Monsterra::All Contract

SECURICHAIN

CONCLUSION

This document, and its appendices, represent our best effort to capture the results of several days of intensive activity.

Smart contracts within the scope were analyzed with static analysis tools and manually reviewed.

Please feel free to direct any questions on this assessment to: audit@securichain.io

SECURICHAIN

APPENDIX 1: ASSESSMENT LIST

	CHECKLIST	
	Integer Overflow/Underflow	Integer Overflow/Underflow
Arithmetic operations	Integer Truncation	Integer Sign
	Wrong Operator	
Re-entrancy		
Bad Randomness	Timestamp Dependence	Blockhash
Front running		
DDos	DOS By Complex Fallback Function	DOS By Gaslimit
	DOS By Non-existent Address Or Malicious Contract	
Gas usage	Invariants in Loop	Invariants State Variables Are Not Declared Constant
Unsafe external calls		
Business Logics Review		
Access Control & Authorization	Replay Attack	Use tx.origin For Authentication
Logic Vulnerability		

SECURICHAIN

APPENDIX 2: LIST RATING

Risk Level	Explain	Example Types
High	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.	Re-entrancy Front running DDos Bad Randomness Logic Vulnerability Arithmetic operations
Medium	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.	Access Control Unsafe external calls Business Logics Review Logic Vulnerability
Low	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.	Gas Usage
Info	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth.	Blockhash