SECURICHAIN

SMART CONTRACT AUDIT

SECURITY ANALYSIS REPORT FOR

EPIC WAR

August 2nd , 2022

SECURICHAIN

Security Rating

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

Epic War

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.

Epic War

SECURICHAIN

TABLE OF CONTENTS

Epic War

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	**EPIC WARS**
Platform	**Ethereum**
Languages	**Solidity**
Methods	**Automation scan, architecture review, functional testing, manual code review**
Repository	**https://github.com/Epic-Wars/game-contract/tree/0d2cdbce730edb6541c1f16e476de902a752edf3 https://github.com/Epic-Wars/mystery-box-contract/tree/b320a403ad035a6cebf809fe4e0ad896bf85017a**
Documents	
Timelines	**May 6th 2022 – Jun 15th 2022**

SECURICHAIN

1.3. CHECKSUM FILE

SCOPE

No.	Hash	Name
1	3bf3bcc5fcc8a9d7c555ae99d127f461a2aea5e8	mystery-box-contract/ EpicWarNFT.sol
2	bd01a782a3e71224ea4d4f3db7706d383ec8b545	mystery-box-contract/ EpicWarBox.sol
3	451d1db90af5f770e7bdfc23fd5beaf77eaea25e	mystery-box-contract/ EpicWarNumber.sol
4	fe63ef5eb4661c86a5744c3f91db6fb80a7f7852	mystery-box-contract/ TokenTest.sol
5	a3a4e0295c29a1a5676290510ca6a02c5f8b0faa	game-contract /EpicWarNFTTest.sol
6	bb2dd76250631963aef8c834aa69d56c84f3f3eb	game-contract /NFTEscrow.sol
7	fe63ef5eb4661c86a5744c3f91db6fb80a7f7852	game-contract /TokenTest.sol
8	e7ab4afe642260ac6c92575e46a08a13620f3513	game-contract /TransferHelper.sol
9	b9a728eb3d6eac741d08ea45fda3fe54494806fc	game-contract /interface/INFTEpicWar.sol
10	9f1c433c84ac01e4def589a49a85e43cdce3a957	game-contract /interface/INFTEscrow.sol

1.4. ASSESSMENT RESULTS

According to the assessment, the Customer's smart contracts have a security rating of 95/100

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	Unauthorized execution	1	Resolved
SC2	Medium	Re-entrancy	1	Resolved
SC3	Medium	Business Logic	1	Acknowledged
SC5	Low	Unchecked condition	1	Resolved
SC6	Low	Gas optimization	5	Resolved
SC7	Information	Unlocked Pragma		Resolved

For rating each vulnerability, refer to Appendix 2.

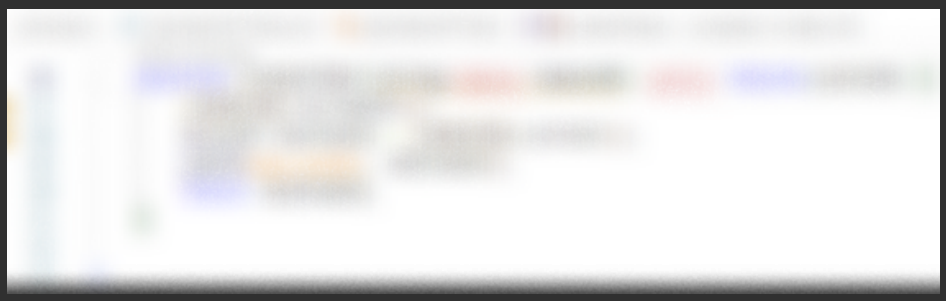
SECURICHAIN

[1] Unauthorized execution

High: 1

Overview

There is no check on the authorization of the call to “createToken” function which leads to unauthorization execution.



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

Posible Impacts

Anyone can mint new token which leads to the loss of its value.

Recommendation

Add “onlyOwner” modifier to the function implementation

Location

Game-contract/EpicWarNFTTest.createToken() #L46

SECURICHAIN

2.1 Details

[2] Re-entrancy

Medium: 1

Overview

Function buyBox() calls function _safeMint() to mint a new box (line 209):

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

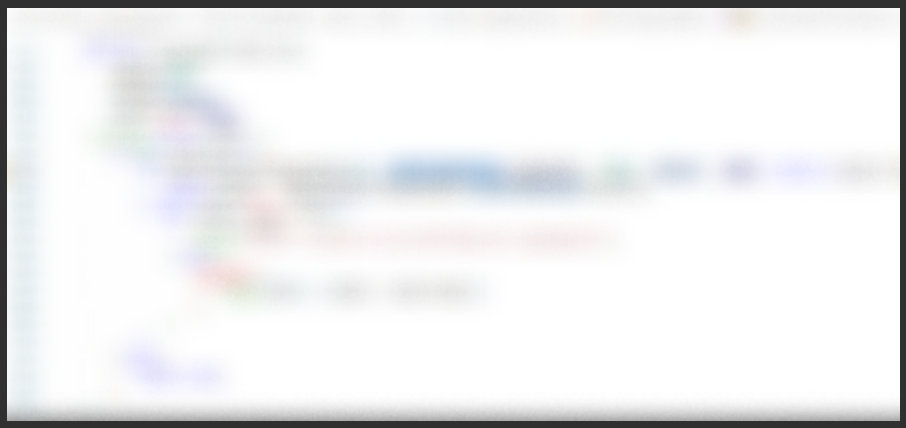
Then function _safeMint() calls private function _checkOnERC721Received() to check if the box was successfully minted:

SECURICHAIN

2.1 Details

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

Inside _checkOnERC721Received() function, a call to _to.onERC721Received() is made:



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

The problem is, the msg.sender contract might have a function called onERC721Received() implemented which re-call EpicWarBox.buyBox() function, causing re-entrancy state.

Possible Impacts

The attacker can buy more boxes than he is allowed.

Recommendation

Use "_mint" instead of "_safeMint".

Location

Mystery-box-contract/EpicWarBox.buyBox() #L209

SECURICHAIN

2.1 Details

[3] Business Logic

Medium: 1

Overview

No detailed description according to customer’s request.

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

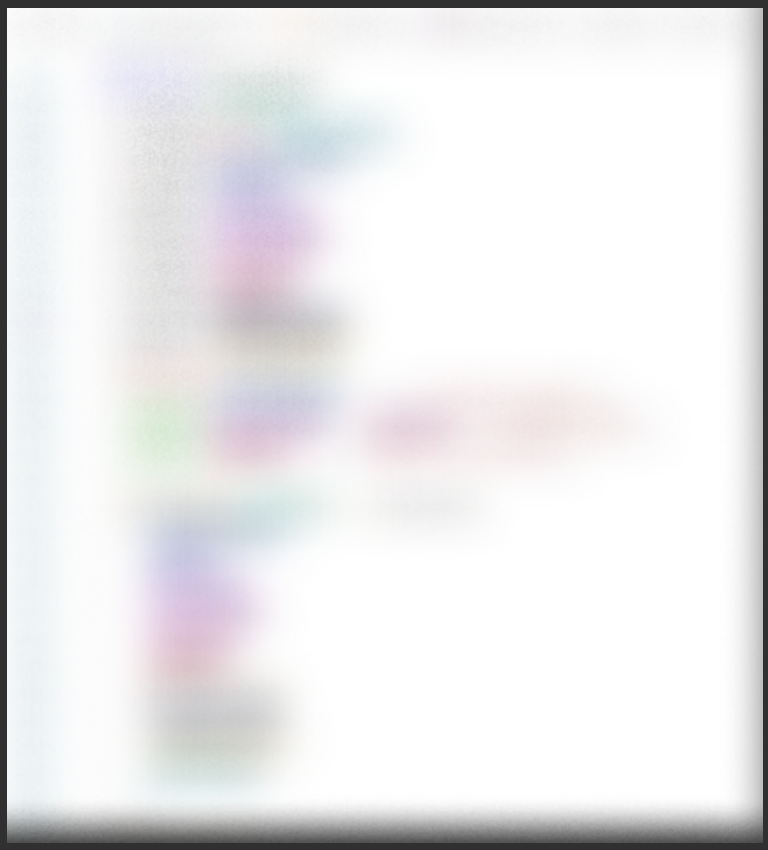
SECURICHAIN

[4] Unchecked condition

Medium: 1

Overview

Function createEvent() does not check for the validity of variable _openBoxTime's value, e.g. its value can be less than startTime's.



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

Possible Impacts

New eventInfo' misconfigured variables can lead to unexpected behaviors of the contract

Recommendation

Add condition-checking requirements.

Location

Mystery-box-contract/EpicWarBox.createEvent() #L117-158

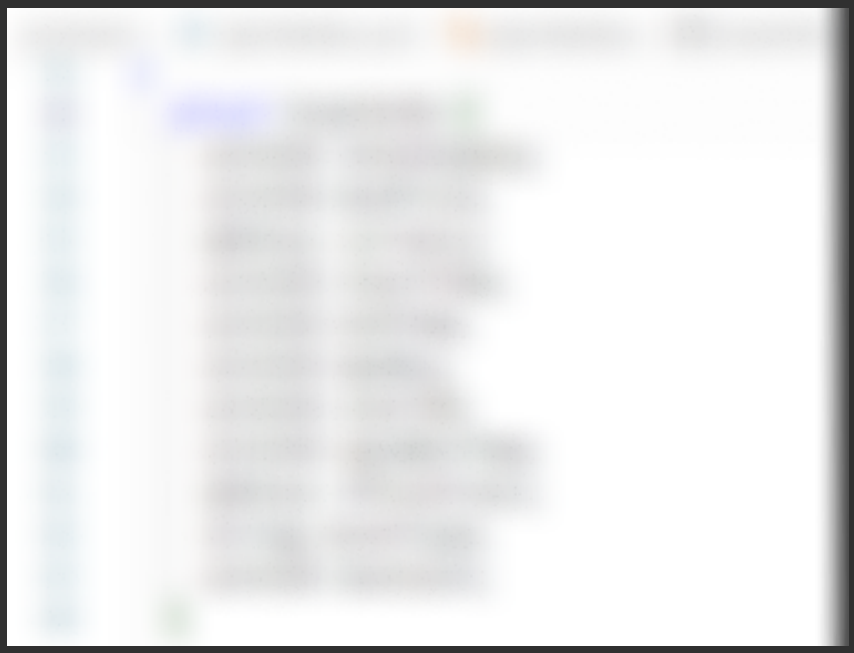
SECURICHAIN

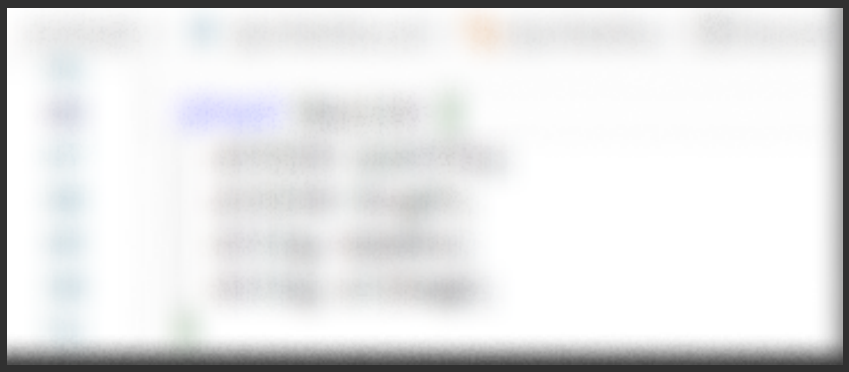
[5] Gas Optimization

Medium: 5

Overview

Some struct fields are implemented using unnecessarily large data type.





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

Struct fields:
- EventInfo.boxPrice
- EventInfo.maxBuy
- EventInfo.boxCount
- BoxList.quantity
- BoxList.bought

Theoretically, those values can be that large, but in reality we can save gas usage by using smaller data type.

Possible Impacts

Gas is wasted to verify a transaction.

Recommendation

Use smaller data type like uint32, uint 64, uint128 depending on their estimated values.

Location

Mystery-box-contract/EpicWarBox.EventInfo.boxPrice #L34

Mystery-box-contract/EpicWarBox.EventInfo.maxBuy #L38

Mystery-box-contract/EpicWarBox.EventInfo.boxCount #L43

Mystery-box-contract/EpicWarBox.BoxList.quantity #L47

Mystery-box-contract/EpicWarBox.BoxList.bought #L48

SECURICHAIN

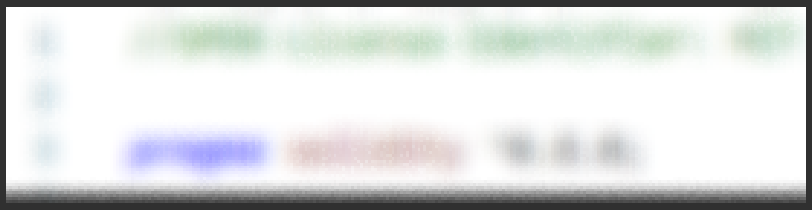
[6] Unlocked Pragma

Information: 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.

Possible Impacts



(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 with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

Location

Mystery-box-contract:: All Contracts

Game-contract:: All Contracts

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	
Arithmetic operations	Integer Overflow/Underflow	Integer Overflow/Underflow
	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