SECURICHAIN

SMART CONTRACT AUDIT

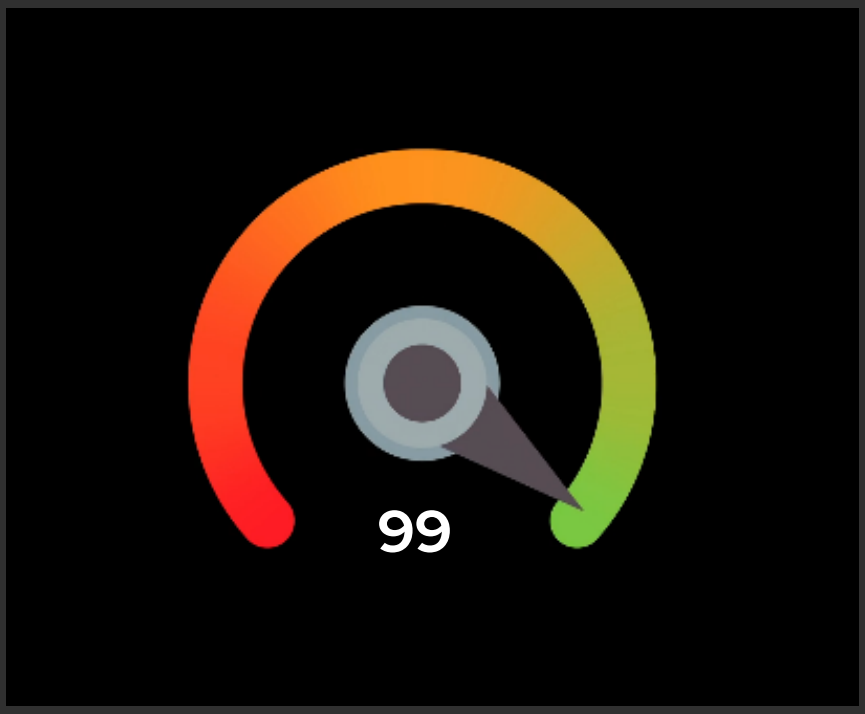
SECURITY ANALYSIS REPORT
FOR

DECENTRALIZED ESPORTS (DES)

May 30th , 2022

SECURICHAIN

Security Rating



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

DES

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.

DES

SECURICHAIN

TABLE OF CONTENTS

DES

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	**Decentralized ESports**
Platform	**Ethereum**
Languages	**Solidity**
Methods	**Automation scan, architecture review, functional testing, manual code review**
Repository	**https://github.com/decentralized-esports/smartcontracts/tree/919dbe419336540c12fcc26a3d8ea4e4715eaa32**
Documents	
Timelines	**May 23th, 2022 – May 30th, 2022**

SECURICHAIN

1.3. CHECKSUM FILE

SCOPE

No.	Hash	Name
1	fc135a2fad7a46ca735e9c3826a700b3dcffafc0	DecentralizeESportToken.sol
2	fc76286936ee15989a2bb91d79ef12603ad1c9bf	EventEmitter.sol
3	3141ebbc9d1092158a2330448d0db97b1781eef0	Ladder.sol
4	ada697aea74b0ced03f210786f8b725d3dc5de42	Tournament.sol
5	11851c26241f50f5c4fa27ba4350ef47ebb190e5	TournamentFactory.sol

SECURICHAIN

1.4. ASSESSMENT RESULTS

According to the assessment, the Customer's smart contracts have a security rating of 99/100

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	Re-entrancy	1	Resolved
SC2	Medium	Arithmetic operations	2	Resolved
SC3	Medium	Uninitialized index variable	3	Resolved
SC4	Medium	Unchecked zero address	1	Resolved
SC5	Low	Gas optimization	1	Resolved
SC6	Information	Unlocked Pragma	1	Resolved

For rating each vulnerability, refer to Appendix 2.

SECURICHAIN

2.1 Details

[1] Re-entrancy

High: 1

Overview

The contract removes reward-claiming address “msg.sender” from “userRewards” after transferring money. This is dangerous as “msg.sender” might contain a callback function implemented which re-call “claimRewards” whenever a money-transferring event occurs.

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

Possible Impacts

The attacker can withdraw all money in this contract.

Recommendation

Remove “msg.sender” from “userRewards” before making money-transferring call.

Location

Ladder.claimRewards() #L367

SECURICHAIN

[2] Arithmetic operation

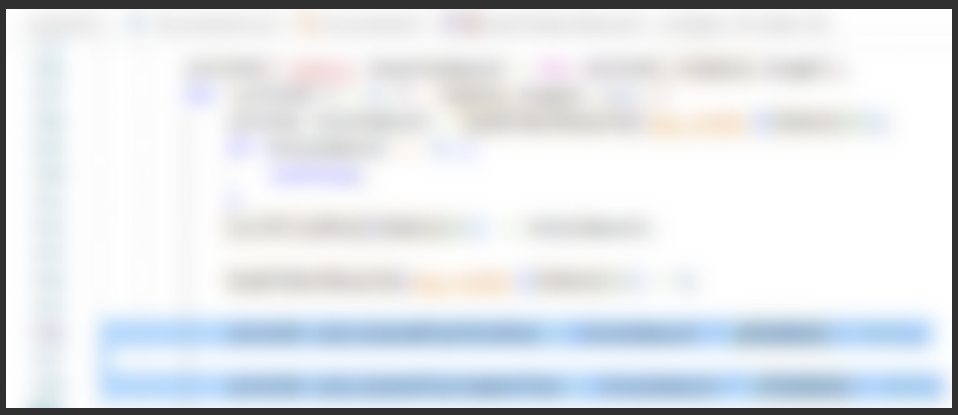
Medium: 2

Overview

Multiplication is performed after division which can lead to unwanted result

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

DES



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

“pfeeRate” and “tfeeRate” are divided by 100 while their value falls between 0 and 100; therefore, the result of the division is always zero.

Impossible Impacts

Platform and tournament will not receive withdraw fee and reward-claiming fee.

Recommendation

Perform multiplication before division.

Location

Tournament.withDrawTournament() #L391-#395
Tournament.withDrawTournament() #L736-#738

DES

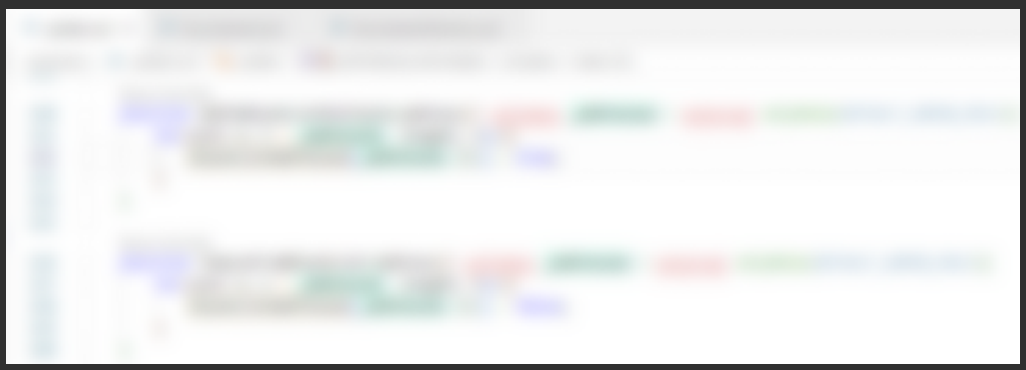
SECURICHAIN

[3] Uninitialized index variable

Medium: 3

Overview

Local variables should be initialized before being used.



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

Impossible Impacts

Contract might perform unwanted behaviors.

Recommendation

Initialize variables before using.

Location

Ladder.addToBlackListMultiple() #L151
Ladder.removeFromBlackList() #L157
Tournament.addToBlackListMultiple() #L486

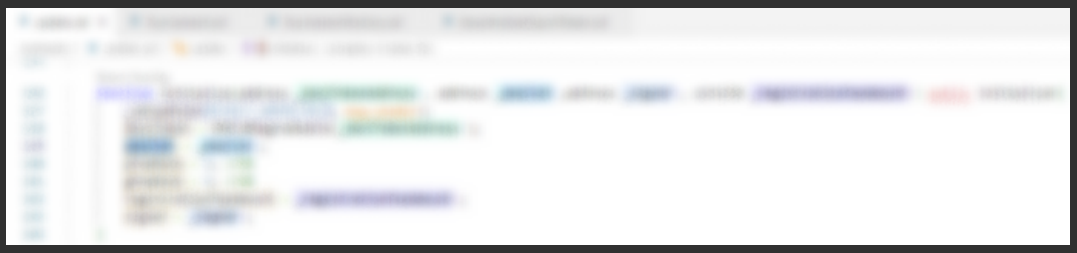
SECURICHAIN

[4] Unchecked zero address

Medium: 1

Overview

“pWallet” variable is not checked for zero value



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

Impossible Impacts

Contract might perform unexpected behaviors

Recommendation

Check variable “pWallet” for zero value.

Location

Ladder.initialize() #L139

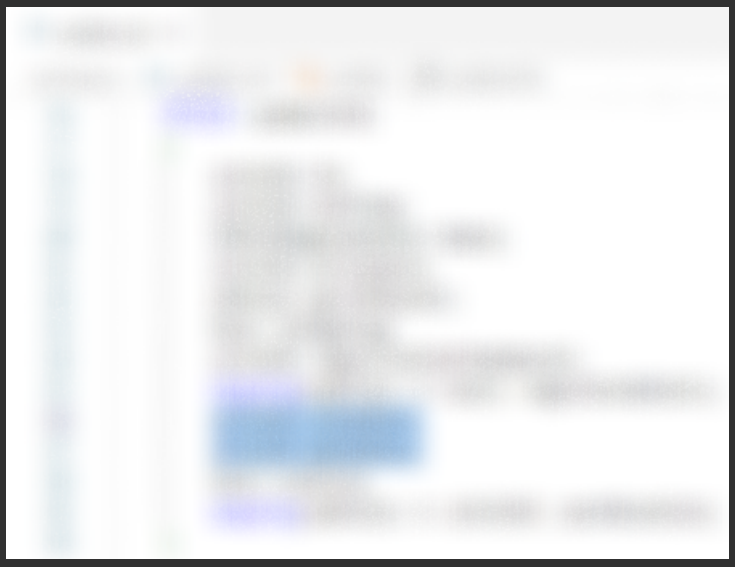
SECURICHAIN

[5] Gas Optimization

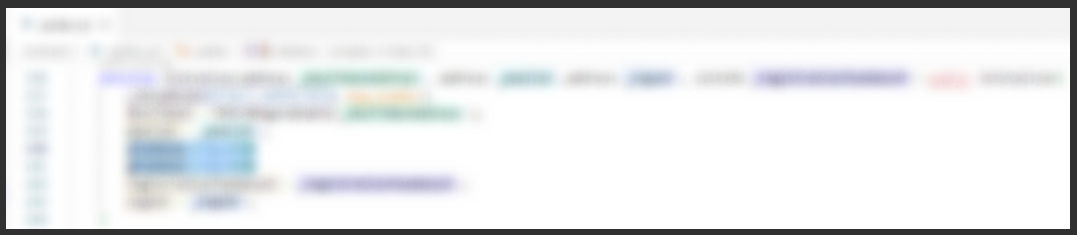
Low: 1

Overview

Struct fields are implemented using unnecessarily large data type.



The reason is that “pFeeRate” and “gFeeRate” contain numbers that are less than 100 and more than 0.



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

Posible Impacts

Gas is wasted to verify a transaction.

Recommendation

Use type “uint8”.

Location

Ladder.LadderInfo #L86,87

SECURICHAIN

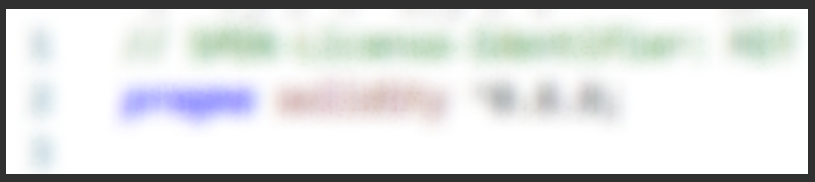
[6] Unlocked Pragma

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.

Posible Impacts



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 in order to compile locally.

Location

DES:: All Contract
DES:: 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

DES

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