SECURICHAIN

SMART CONTRACT AUDIT

SECURITY ANALYSIS REPORT FOR

ISLANDER

May 10th , 2022

SECURICHAIN

Security Rating

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

Islander

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.

Islander

SECURICHAIN

TABLE OF CONTENTS

Islander

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	**Islander**
Platform	**ERC20**
Languages	**Solidity**
Methods	**Automation scan, architecture review, functional testing, manual code review**
Repository	**URL: https://github.com/Spiderum/islander-contracts Commit: 2873eb0625b7e47190942f4d0df531edcd72c985**
Documents	
Timelines	**April 18th, 2022 – April 26th, 2022**

SECURICHAIN

1.3. CHECKSUM FILE

SCOPE

No.	Hash	Name
1	f31915e54fcf5d267edcc6a54b3022860334364ca7b511362b7738849bbde2d7	ISAStaking.sol
2	76df8d880bc6d0e3c97623249c8624441f88b6cff3e86ef3935d8d832df990fa	QuestFundV2.sol
3	7ba2f99340150df3297da12e800fd431697cea603d4635f71f464a7ea7ff66c0	QuestV4.sol
4	a9a1d1f294808329003460ffb098c4e4693200f1dbec03117b510c54f2ac3989	VerifySignature.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

Islander

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	Information	Unlocked Pragma	1	Resolved
SC2	Low	Gas Optimization	1	Acknowledged
SC3	Low	untrustedClaimReward() - wrong behavior	1	Acknowledged
SC4	Low	Missing Zero Address Validation	1	Acknowledged
SC5	Low	Unchecked Return Value	1	Resolved

For rating each vulnerability, refer to Appendix 2.

SECURICHAIN

2.1 Details

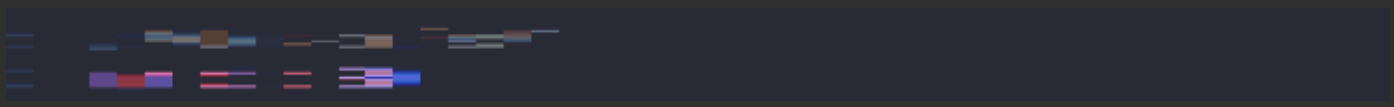
[1] Unlocked Pragma

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 might introduce bugs affecting 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

Islander: All Contracts

SECURICHAIN

[2] Gas Optimization

Low: 1

Overview

Gas optimization is a matter of doing what is cheap and avoiding what is expensive in terms of gas costs on EVM blockchains.

Impossible Impacts

Variable `totalRewards` is declared and assigned a value but never used.

Recommendation

Delete variables if not in use.

Location

ISAStaking#L33

ISAStaking.fund()

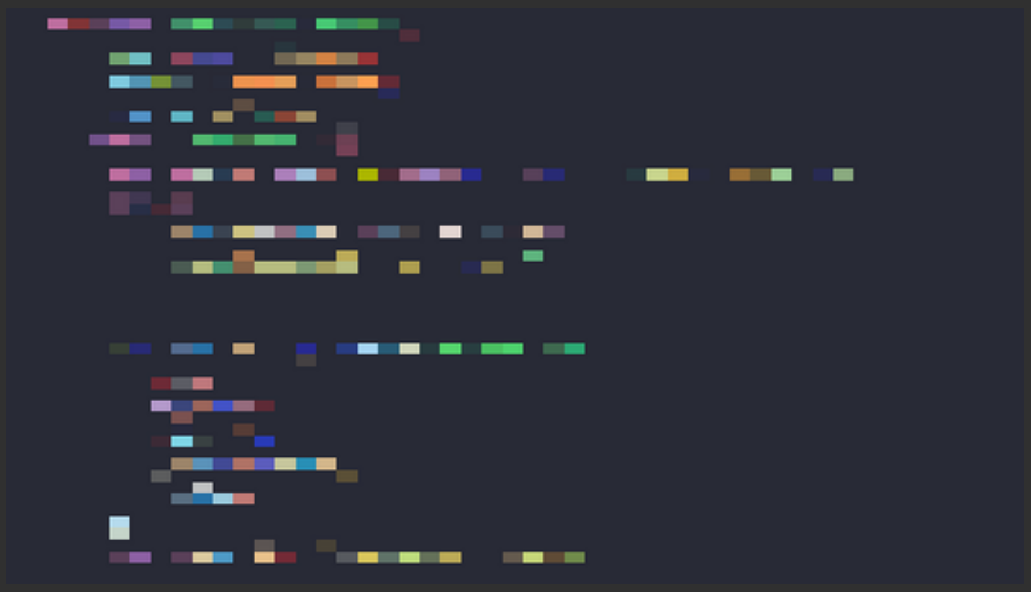
SECURICHAIN

[3] untrustedClaimReward() behaved wrongly

Low: 1

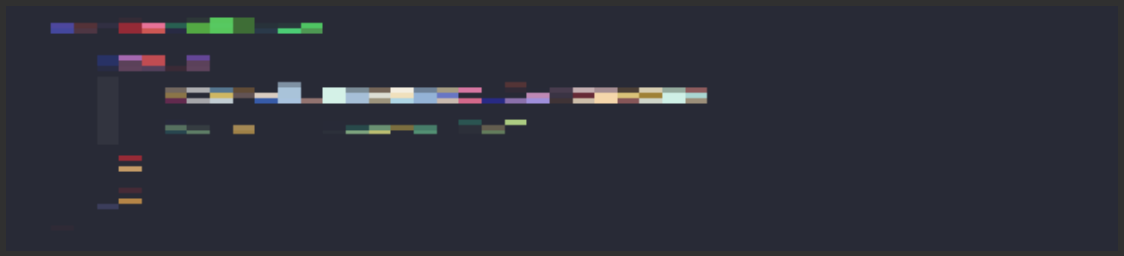
Overview

`Islander` documentation says: "Notice that `untrustedClaimReward` can only be called when the time is not over `expiredTimestamp`". But untrustedClaimReward() was checked `withdrawableTimestamp` via ` withdrawable` modifile.



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

` withdrawable` modifile:



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

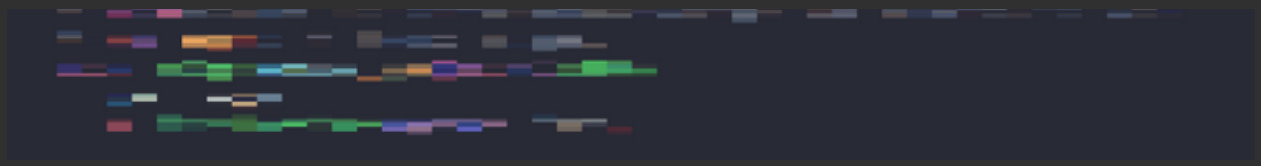
SECURICHAIN

[4] Missing Zero Address Validation

Low: 1

Overview

setSigner() function takes one parameter of type `address` but does not check for an address other than address(0).



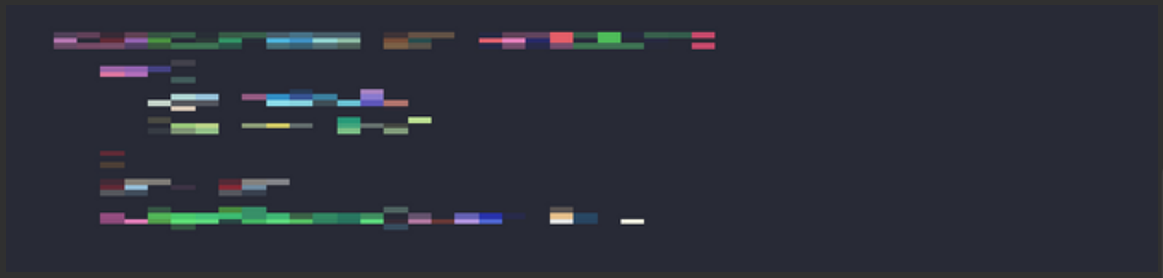
(Blurred image of the code snippet in the public report due to the Customer's code being in the private repository)

Posible Impacts

The owner calls setSigner() without specifying the _signer, so the owner loses signing permission for the quest.

Recommendation

Add `require` to verify _singer other than 0.



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

Location

QuestV4.setSigner()#L145-148

SECURICHAIN

[5] Unchecked Return Value

Low: 1

Overview

The result returned from the `transfer` process is not rechecked (success or failure)

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

`payOut` will add ` _amount` even if `transfer()` fails.

Recommendation

Use safeTransfer() of SafeERC20Upgradable.sol

Location

ISAStaking.erc20Transfer()#L280

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