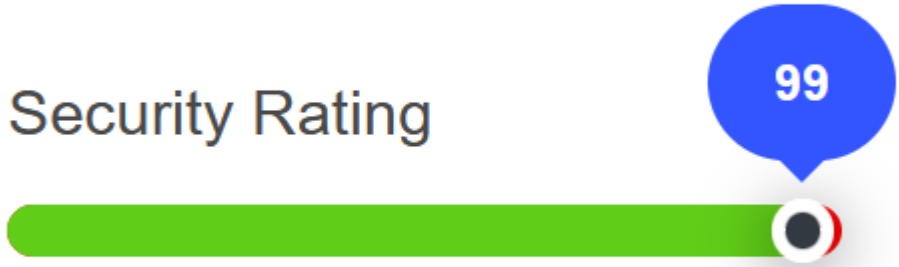
SECURICHAIN

SMART CONTRACT SECURITY ANALYSIS REPORT FOR

GAMEFI

Feb 7th 2022



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

Disclaimer

This report contains 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 provide investment advice, nor should be leveraged as investment advice of any sort.

TABLE OF CONTENTS

1. 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 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.

1.2. SCOPE OF WORK

Project Name	GAMEFI
Platform	ETHEREUM
Languages	SOLIDITY
Methods	AUTOMATION SCAN, ARCHITECTURE REVIEW, FUNCTIONAL TESTING, MANUAL CODE REVIEW
Repository	STAKING: COMMIT C74C00C MARKETPLACE: COMMIT
Documents	
Timelines	**JAN 24TH 2022 – FEB 7TH 2022**

1.3. CHECKSUM FILE

GAFI – MARKETPLACE

No.	Hash	Name
1	ba951481b4cadd9ec674e03c336f4bed02f40032fa983cf5f14fdd8e47023dae	Marketplace.sol
2	381fcc83f64731f4f9e7c2b35190dfe2d85812256235a68d824b38924970300a	IMarketplace.sol
3	329158c21fbbe6631505df67db5f97c61c2a32187e1e09850f90a930ca8f3c73	IStakingContract.sol

GAFI – STAKING

No.	Hash	Name
1	178637867c82180ef823c1f069a259340dd3984f7a2944c236a43e2888cbd223	AllocationPool.sol
2	94f8373214745b366d28a5973b87a975d2253bd8a87b66d1a30b99638ec96f11	LegendNFT.sol
3	7b082cdf44d6d8a9a24c457f855298a4806548655334c6015538defd95434b6c	LinearPool.sol
4	38f2cdf22f9b6e23aa976073190406f59f1ffc18fa1e5c5f2363f219b883b1bb	StakingPool.sol
5	c2ebd8a6011b1fca46df37ea5b9d0c428af0f00c0f1c34a1dc7ee5911a1cc022	ERC20Mock.sol

1.4. ASSESSMENT RESULTS

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

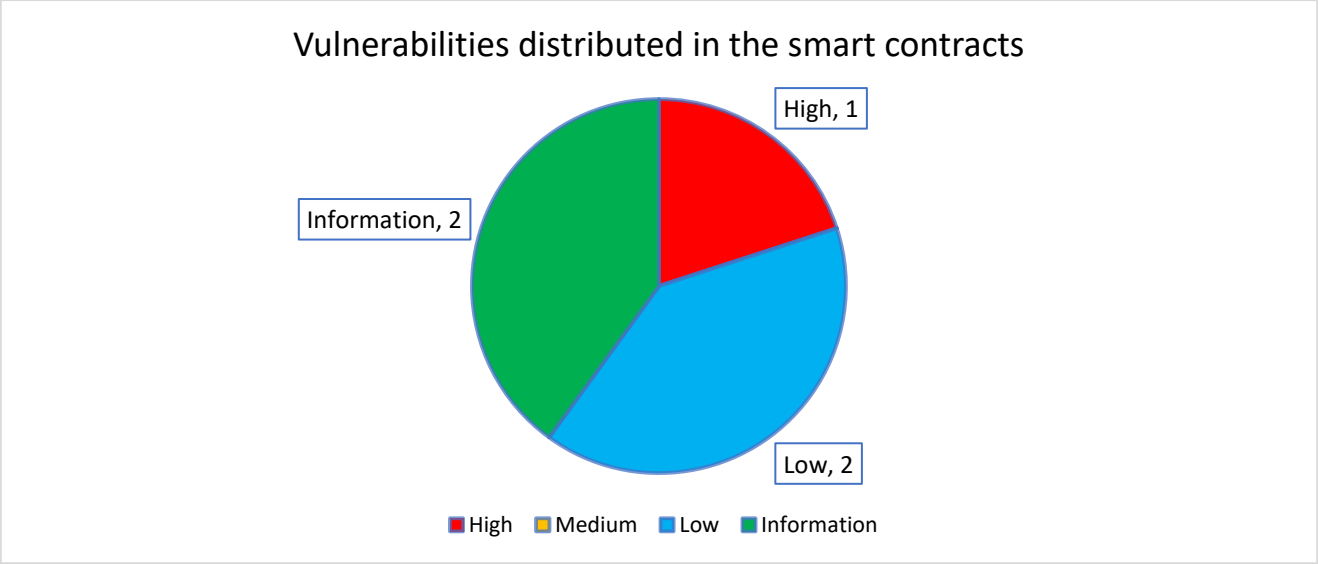
Rate	Description
96-100	**No vulnerabilities** were found or all detected ones have been resolved
70-95	Unresolved **Low-level** vulnerabilities exist
40-69	Unresolved **Medium-level** vulnerabilities exist
0-39	Unresolved **High-level** vulnerabilities exist

(*For information on criteria for risk rating, refer to Appendix.2)*

2. FINDINGS

2.2. LIST OF VULNERABILITIES

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



ID	Risk Level	Name	Amount	**Status** *(after re-checking)*
SC1	Information	Unlocked Pragma	2	Resolved in #cd09c1 commit
SC2	Low	Gas Optimization	2	Resolved in #cd09c1 commit
SC3	High	DoS with (unexpected) revert	1	Resolved in #9baaae commit

(For rating of each vulnerability, refer to Appendix 2.)

2.3. DETAILS

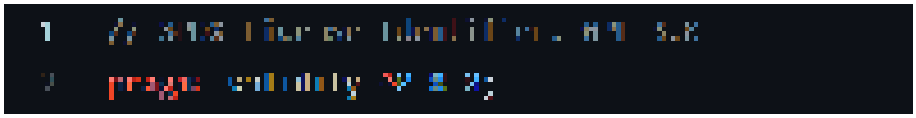
[1] Unlocked Pragma

2 INFO

- **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 Impact**



(Blurring the image of the code snippet in the public report because the Customer's code is 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:**
  - Staking:: ALL CONTRACT
  - Marketplace:: ALL CONTRACT

[2] Gas Optimization

2 LOW

- **Overview**

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

- **Possible Impact**

(Blurring the image of the code snippet in the public report because the Customer's code is in the private repository)

Users have to pay more gas for their requests.

- **Recommendation**

Use ‘external’ instead of ‘public’ for functions that are only called outside of the contract.

- **Location:**
  - Marketplace::Marketplace.sol (#L85 , #L334)

[3] DoS with (unexpected) revert

1 HIGH

- **Overview**

The attacker can win the auction with the smallest price.

- **Possible Impact**

(Blurring the image of the code snippet in the public report because the Customer's code is in the private repository)

1. The attacker first writes a contract to bid on.
2. When someone bids higher, the Contract will return the money to attacker
3. When the funds are returned, the attacker's fallback() function will call revert() causing the transaction to fail
4. Since the transaction that returned the funds to the attacker was faulty, other users can not bid higher.

- **Recommendation**

In view of the above situation, if the result of the external function call needs to be processed before entering the new state, it must be considered that the external call might fail anytime.

- **Location:**
  - Marketplace.sol: (L110 - L136)

3. CONCLUSION

This document, and its appendices, represents the results of several days of our intensive work.

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.

APPENDIX 1. ASSESSMENT LIST

CHECKLIST		
Arithmetic operations		
	Integer Overflow/Underflow	Integer Division
	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	
Unsafe external calls		
Gas usage		
	Invariants in Loop	Invariants State Variables Are Not Declared Constant
Business Logics Review		
Access Control & Authorization		
	Replay Attack	Use tx.origin For Authentication
Logic Vulnerability		

APPENDIX 2. RISK RATING

Risk Level	Explanation	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 Defence in Depth.	Do not specify a specific version of Solidity

