RD AUDITORS

MINT V2 Smart Contract, Code Review and Security Analysis Report

Customer: RPS League
Prepared on: 19th Dec 2021
Platform: Binance Smart Chain
Language: Solidity

rdauditors.com

Table of Contents

Disclaimer

This document may contain confidential information about its systems and intellectual property of the customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the customer or it can be disclosed publicly after all vulnerabilities are fixed - upon the decision of the customer.

Document

Name	Smart Contract Code Review and Security Analysis Report of RPS League
Platform	BSC / Solidity
File 1	MintableBox.sol
MD5 hash	223F09901BD3E70DF37E39DC17C53169
SHA256 hash	0A283319C63621FAD7A75AE4AB94AA2A080A42A9433937F42A9E2AB7C8CA1EF1
File 2	MintableBoxConfigHandler.sol
MD5 hash	14686AAA0B39186794C18D1F1F98E9F6
SHA256 hash	A8D1C5A371C410B2407B7841A014128025144C87B6501C4D8E9F9FCDF661B69E
File 3	MintableBoxERC721Handler.sol
MD5 hash	EC81FAF78BB0130FEC7BABD141E65000
SHA256 hash	00DCD2E80331DA300EA7C4C77705EE28F3787BDB65C366781EECAA6A72B9AFC2
File 4	MintableBoxMintingHandler.sol

MD5 hash	C87EE50968B1FDFC94AE1B0A15195FBD
SHA256 hash	8B95B1465590F2EB86A80CD9AF335AF353136D818F61B6CF0C576B8459167CA2
File 5	MintableBoxOpeningHandler.sol
MD5 hash	2CEFD9EBF7D473D25EA3A687C006173F
SHA256 hash	C67BCBAEEE7A1BA6EA7AAC150EB87E26C4B16BD0F1150812C5AE90DA8F7B14FF
File 6	PriceOracle.sol
MD5 hash	371B3E274220DF8D523621E6F5B711AA
SHA256 hash	8FF07091E3829AF617C67D925240D81221191076D3714B0A6C2538ED55528CDA
File 7	PriceOracleConstants.sol
MD5 hash	15C601DB96E0EF700057F463F61E8436
SHA256 hash	CA3D816AFCEAF586BAC4BA168148A95D90FBA2D7F2531B9901E8FF40DED595A0
File 8	PriceOracleQueriesHandler.sol
MD5 hash	5B009D0CAE989E67796F5FC1C5ECB71E

SHA256 hash	0476BF7DDF148E3EE18AC15D55745F391F9429C3F22CE906920CD5833191BE16
File 9	PriceOracleUpdateHandler.sol
MD5 hash	94AD1A09F273F61CE64C38391D58D47E
SHA256 hash	0B3DA6DE37304EEC01F9D274C4A4CDD31F4AB8C1DF38B646A211338092914841
File 10	RandomNumberConsumer.sol
MD5 hash	5FEA88388E2BB9A946DC7027A9E078C4
SHA256 hash	A34C6C951BAEF959ED919AFC09CA63FF851BF80922C9BB929B25C590E5C9A867
File 11	RandomNumberGenerator.sol
MD5 hash	E98A2F1C56E321296D1D97CACD0214C4
SHA256 hash	22E6743F47854D5789E17B5D12E4A1B0F8DD424280EA81B0A6E5602FFB28C7FF
File 12	RPSNFT.sol
MD5 hash	0629000B535D5A0B6F0080842A441BCA

SHA256 hash	1F098760E05276F9B8A1D4145F586E8B61EEA7AA6C0901D84871A7A45977767A
File 13	RPSNFTConfigHandler.sol
MD5 hash	662FA07994D3B15457DFD4970FBDEC32
SHA256 hash	A42956B15A935C646C30027B8BC9C9FC646F5FDEE788B15FF9FE829C833986A2
File 14	RPSNFTERC721Handler.sol
MD5 hash	302C447E877F42AEF23C72275BED2507
SHA256 hash	9CA5B2D46971B189B032691B94D01D387D01C90C023415702B4C8E1893917CAF
File 15	RPSNFTMintingHandler.sol
MD5 hash	AC45BC31AEDEF75841F8DF493A79C8C6
SHA256 hash	EC86137F5CDCC3CC72F8AA02F4D3F8DEB6F317FC702857A9B0137F73AE437378
Date	19/12/2021

Introduction

RD Auditors (Consultant) were contracted by RPS League (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s smart contracts and its code review conducted between 18 - 19th December 2021.

This contract consists of fifteen files.

Project Scope

The scope of the project is a smart contract. We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is **well-secured.**

Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, the manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found the following;

Total Issues	0
Critical	0
High	0
Medium	0
Low	0
Very Low	0

Code Quality

Please note that within this report safeMath, IERC20, EnumerableSet Math, SafeERC20, ReentrancyGuard, Pausable, Address, ownable are taken from the popular OpenZeppelin library.

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

The RPS League team has not provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is almost not commented on. Commenting can provide rich documentation for functions, return variables and more. Use of Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the RPS League code as a file

The hash of that file is mentioned in the table. As mentioned above, It's recommended to write comments in the smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

RPS MINT V2

File And Function Level Report

File: MintableBox.sol

Contract: MintableBox

Import: MintableBoxOpeningHandler, MintableBoxMintingHandler, IMintableBox

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	supportsInterface	read	Passed	All Passed	No Issue	Passed
2	_beforeTokenTransfer	write	Passed	All Passed	No Issue	Passed

File: MintableBoxConfigHandler.sol

Contract: MintableBoxConfigHandler

Import: Ownable, IMintableBoxConfigHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score

1	wearableChances	read	Passed	All Passed	No Issue	Passed
2	handChances	read	Passed	All Passed	No Issue	Passed
3	setActivationDate	write	Passed	All Passed	No Issue	Passed
4	setRewardPool	write	Passed	All Passed	No Issue	Passed
5	setDevAddress	write	Passed	All Passed	No Issue	Passed
6	setDevFee	write	Passed	All Passed	No Issue	Passed
7	setMintPriceInUSD	write	Passed	All Passed	No Issue	Passed
8	setPriceOracle	write	Passed	All Passed	No Issue	Passed
9	setBaseURI	write	Passed	All Passed	No Issue	Passed
10	setPaymentToken	write	Passed	All Passed	No Issue	Passed
11	setMaxDelay	write	Passed	All Passed	No Issue	Passed
12	setNFTMintingContract	write	Passed	All Passed	No Issue	Passed
13	setWearableChances	write	Passed	All Passed	No Issue	Passed
14	setHandChances	write	Passed	All Passed	No Issue	Passed
15	_setWearableChances	write	Passed	All Passed	No Issue	Passed
16	_setHandChances	write	Passed	All Passed	No Issue	Passed
17	_areStringsTheSame	read	Passed	All Passed	No Issue	Passed

File: MintableBoxERC721Handler.sol

Contract: MintableBoxERC721Handler

Import: MintableBoxConfigHandler, ERC721Enumerable, IMintableBoxERC721Handler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	tokenType	read	Passed	All Passed	No Issue	Passed
2	tokenURI	read	Passed	All Passed	No Issue	Passed
3	_mint	write	Passed	All Passed	No Issue	Passed

File: MintableBoxMintingHandler.sol

Contract: MintableBoxMintingHandler

Import: MintableBoxERC721Handler, IMintableBoxMintingHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	purchaseBoxes	write	Passed	All Passed	No Issue	Passed
2	airdropMany	write	Passed	All Passed	No Issue	Passed
3	airdrop	write	Passed	All Passed	No Issue	Passed
4	_getPriceInToken	read	Passed	All Passed	No Issue	Passed

File: MintableBoxOpeningHandler.sol

Contract: MintableBoxOpeningHandler

Import: MintableBoxERC721Handler, RandomNumberConsumer, IMintableBoxOpeningHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	openBoxes	write	Passed	All Passed	No Issue	Passed
2	replayBoxOpenings	write	Passed	All Passed	No Issue	Passed
3	supportsInterface	read	Passed	All Passed	No Issue	Passed
4	_consumeRandomNumber	write	Passed	All Passed	No Issue	Passed
5	_beforeTokenTransfer	write	Passed	All Passed	No Issue	Passed
6	_pickOneOfChances	read	Passed	All Passed	No Issue	Passed
7	_getRandomWearable	write	Passed	All Passed	No Issue	Passed
8	_getRandomHands	write	Passed	All Passed	No Issue	Passed
9	_calculateNextTokenId	write	Passed	All Passed	No Issue	Passed
10	_generateMultipleNumbers	read	Passed	All Passed	No Issue	Passed

File: PriceOracle.sol

Contract: PriceOracle

Import: PriceOracleConstants, PriceOracleQueriesHandler, IPriceOracle

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	decimals	read	Passed	All Passed	No Issue	Passed
2	description	read	Passed	All Passed	No Issue	Passed
3	version	read	Passed	All Passed	No Issue	Passed
4	getRoundData	read	Passed	All Passed	No Issue	Passed
5	latestRoundData	read	Passed	All Passed	No Issue	Passed

File: PriceOracleConstants.sol

Contract: PriceOracleConstants

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	decimals	read	Passed	All Passed	No Issue	Passed
2	description	read	Passed	All Passed	No Issue	Passed
3	version	read	Passed	All Passed	No Issue	Passed

File: PriceOracleQueriesHandler.sol

Contract: PriceOracleQueriesHandler

Import: PriceOracleUpdateHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	getRoundData	read	Passed	All Passed	No Issue	Passed
2	latestRoundData	read	Passed	All Passed	No Issue	Passed

File: PriceOracleUpdateHandler.sol

Contract: PriceOracleUpdateHandler

Import: Ownable, IPriceOracleUpdateHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	canAddressUpdate	read	Passed	All Passed	No Issue	Passed
2	setAddressPermission	write	Passed	All Passed	No Issue	Passed
3	updatePrice	write	Passed	All Passed	No Issue	Passed

File: RandomNumberConsumer.sol

Contract: RandomNumberConsumer

Import: Ownable, ERC165, IRandomNumberConsumer

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setGenerator	write	Passed	All Passed	No Issue	Passed
2	supportsInterface	read	Passed	All Passed	No Issue	Passed
3	consumeRandomNumber	write	Passed	All Passed	No Issue	Passed
4	_requestRandomNumber	write	Passed	All Passed	No Issue	Passed

File: RandomNumberGenerator.sol

Contract: RandomNumberGenerator

inherits: VRFConsumerBase, Ownable, IRandomNumberGenerator

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	requestRandomNumber	write	Passed	All Passed	No Issue	Passed
2	setAddressesPermissions	write	Passed	All Passed	No Issue	Passed
3	withdrawLink	write	Passed	All Passed	No Issue	Passed

4	_isAddressConsumer	read	Passed	All Passed	No Issue	Passed
5	fulfillRandomness	write	Passed	All Passed	No Issue	Passed

File: RPSNFT.sol

Contract: RPSNFT

inherits: RPSNFTMintingHandler, IRPSNFT

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	constructor	write	Passed	All Passed	No Issue	Passed

File: RPSNFTConfigHandler.sol

Contract: RPSNFTConfigHandler

inherits: Ownable, IRPSNFTConfigHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setBaseURI	write	Passed	All Passed	No Issue	Passed
2	setAddressPermission	write	Passed	All Passed	No Issue	Passed

File: RPSNFTERC721Handler.sol

Contract: RPSNFTERC721Handler

inherits: RPSNFTConfigHandler, ERC721, IRPSNFTERC721Handler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	tokenURI	read	Passed	All Passed	No Issue	Passed

File: RPSNFTMintingHandler.sol

Contract: RPSNFTMintingHandler

inherits: RPSNFTERC721Handler, IRPSNFTMintingHandler

Observation: Passed

Test Report: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	mint	read	Passed	All Passed	No Issue	Passed

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Critical:

No critical severity vulnerabilities were found.

High:

No high severity vulnerabilities were found.

Medium:

No medium severity vulnerabilities were found.

Low:

No low severity vulnerabilities were found.

Very Low:

No very low severity vulnerabilities were found.

Discussion:

1. Check the static values before deployment.
2. Loops may fail/consume high gas depending upon iterator length.

Conclusion

We were given a contract file and have used all possible tests based on the given object. The contract is written systematically, so it is ready to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is now “well secured”

Note For Contract Users

Owner has full control over the smart contract. Thus, technical auditing does not guarantee the project's ethical side.

Please do your due diligence before investing. Our audit report is never an investment advice.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

RD Auditors Disclaimer

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Email: info@rdauditors.com

Website: www.rdauditors.com