verichains

SECURITY AUDIT OF

RONIN MARKETPLACE CONTRACTS

Public Report

Mar 18, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Mar 18, 2024. We would like to thank the Ronin for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the Ronin Marketplace contracts. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team found no vulnerabilities in the given version of Ronin Marketplace contracts, only some notes and recommendations.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Ronin Marketplace contracts

Ronin is an EVM blockchain specifically forged for gaming. Launched by Sky Mavis, the creator of Web3's breakout title Axie Infinity which has generated over $1.3 B in revenue, Ronin is the only blockchain proven to scale a single game to accommodate millions of daily active users and has processed over $4 B in NFT volumes. Ronin optimizes for near-instant transactions and negligible fees that enable millions of in-game transactions to occur seamlessly, making it the leading choice for Web3 games.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of Ronin Marketplace contracts. It was conducted on the following contracts on the Ronin Chain:

#	Contract	Address
1	Market Gateway Proxy	0x3b3adf1422f84254b7fbb0e7ca62bd0865133fe3
2	Marketplace Gateway V2	0xfff9ce5f71ca6178d3beecedb61e7eff1602950e
3	Market Gateway Logic	0x53a11a02195d284c78b79801056ea893a2e6ea09
4	Mavis Order Exchange	0xB24592970973a80014fa80Feb895873094909B3C
5	App Axie Order Exchange	0xe35CbC0a0F2025E3bD9ec8e1f30644Df333C820f
6	ERC1155 Exchange	0xb36C9027eD4353fdD7A59d8c40E0dF5221a3764F
7	Market Gateway Multi Send	0x5404eba4bc27b49a66b7fadc62dd17359dce1de1

Note that `Market Gateway Proxy` and `Marketplace Gateway V2` are upgradable proxy contracts which are pointed to implementationat `Market Gateway Logic` above. These contracts can be upgraded to other implements in thefuture, so we only audit the current implementation as of the report writing time.

The Ronin Marketplace contracts use `MarketCommission` contract to calculate market commission and `KatanaRouter` to swap tokens which are out of scopein this audit. `WRON` and `WRONHelper` which are used to wrap and transfer native tokens are also out of scope.

The following files were made available in the course of the review:

SHA256 Sum	File
0a14fd07f74ac37d9bfbaaceda8f7b401c80c014e5925b6c6b86db9357b40d9e	LibGenericOrder.sol
daeddfab5d10b0eb0d402e781b76f2327fda7c8ce2662d5547931aa5d5b45043	LibOrder.sol
d59dcdb042da3824b03db4d0c4a04dce35fdc9ae2a3222a333c44fc799e503f4	LibERC1155Order.sol
dd7b03eb573da80f944df7e8a8db50034987f153c92b8b02a6d3ef88bd4d8c1c	LibAsset.sol
b5e24e8ba517e1bf4cd5fbe6040d7329971f0c038f11c3463e30a086e1c9c9ab	Convert.sol
e013c492c04d47b13329543a5060072f5cd6a12d5a50d2805aacbf64bd690813	SharedGatewayFragment.sol
04e1ae3be652e281698801ab816f512f6ad586af7b79c4f4435d94c54d555c5c	TransferFromHelper.sol
a849a195d41e6eff9cd44377237953d9bfc3f7b766331934d4049c60f045f4e8	RONTransferHelper.sol
3afce4972efb6d056297e5482c30f64b2e19ecf4dacc36a33f4fd3a6e594c876	TransferHelper.sol
9945431e3c020321c2abe95338c5dc516aff6cc72c264d8f38d15c628fa2bdfb	RONTransferHelperExtended.sol
54345b34e2e301c9d6678d54fdf4168a2056cc75103c4ec942caa2b1b2ae011b	CoreExchange.sol
3a25c38ae74d49006128f0ad9700a1d91a547922a52b1827364b051ba7cf6276	ERC1155Exchange.sol
4ca1a2b657543e30e3f7be3cf3c2481ce963e8abe96ed68e7937b07b77e47849	LegacyOrderExchange.sol
10d92911a6a0c6a39905d472ae96c45033b837fe455a5d8b11d1a769b4409217	AppAxieOrderExchange.sol
7d2fac93d713684e859ba1763d67275915085125949764d3d1b9bf98645299e4	MavisOrderExchange.sol

f8b8305602a7538d29bb25f10a1640abc95719f32c563c2e5972dd0350c85b4d	GatewayGetterSetter.sol
6c81a0eb0c59286af21996c9b1cb862e5496bfda54732b633d0d6762199eb9fc	MarketGateway.sol
04e0bb5daa7ca8d52225ec73f28d0b0ac009377e85047555fca1adf3f0d936db	MarketGatewayMultiSend.sol

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Ronin acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. Ronin understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Ronin agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the Ronin will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Ronin, the final report will be considered fully accepted by the Ronin without the signature.

2. AUDIT RESULT

2.1. Overview

The Ronin Marketplace contracts was written in `Solidity` language, with the required version to be `^0.8.0`.

2.1.1. CoreExchange abstract contract

This is the base code for Ronin Marketplace contracts. It provides essential functions for a token marketplace contract, including verifying, settling, and canceling orders. Additionally, the contract allows the use of both native tokens and other supported tokens, enabling swaps to the designated payment token for order settlement.

The marketplace supports place orders and offers by signing signature and the order will be verified using the signature by the contract before being settled.

2.1.2. ERC1155Exchange, LegacyOrderExchange contracts

The `ERC1155Exchange` and `LegacyOrderExchange` contracts inherit from the `CoreExchange` base contract, providing a user-facing interface for token trading. `ERC1155Exchange` is specifically designed for trading `ERC1155` tokens, while `LegacyOrderExchange` supports the trading of `ERC20` and `ERC721` tokens.

2.1.3. AppAxieOrderExchange, MavisOrderExchange contracts

Both `AppAxieOrderExchange` and `MavisOrderExchange` contracts inherit from the `LegacyOrderExchange` contract. The key distinction is that the `MavisOrderExchange` does not support bundle orders.

2.1.4. MarketGateway contract

The MarketGateway contract acts as a central point of interaction for users, allowing them to interact with the `AppAxieOrderExchange`, `MavisOrderExchange` and `ERC1155Exchange` contracts.

2.1.5. MarketGatewayMultiSend contract

This contract enables users to settle multiple orders simultaneously by interacting with supported `MarketGateway` contracts. It supports both `ERC20` and native tokens as payment methods and automatically refunds any excess amount to the sender.

2.2. Findings

During the audit process, the audit team found no vulnerabilities in the given version of Ronin Marketplace contracts, only some notes and recommendations.

2.2.1. Should extend ReentrancyGuardUpgradeable for upgradable contracts INFORMATIVE

Affected files:

- MarketGateway.sol
- MarketGatewayMultiSend.sol

The `MarketGateway` and `MarketGatewayMultiSend` contracts are upgradable and deployed behind a proxy contract. Because they extend `ReentrancyGuard`, the `_status` variable is set redundantly during the implementation contract's construction. This leads to higher gas costs and doesn't properly initialize the `_status` variable in the proxy's storage.

```
abstract contract ReentrancyGuard {
    ...
    uint256 private constant _NOT_ENTERED = 1;
    uint256 private constant _ENTERED = 2;

    uint256 private _status;

    constructor() {
        _status = _NOT_ENTERED;
    }
    ...
}
```

RECOMMENDATION

To optimize the contracts and ensure correct `_status` initialization, follow these steps:

- Modify the `MarketGateway` and `MarketGatewayMultiSend` contracts to extend `ReentrancyGuardUpgradeable` instead of `ReentrancyGuard`.
- Call the `__ReentrancyGuard_init` function from within the proxy contract's initialize function.

The `ReentrancyGuardUpgradeable` contract is specifically designed for use with upgradable contracts and proxies. By calling `__ReentrancyGuard_init` within the proxy's `initialize` function, you ensure that the reentrancy protection status is correctly initialized within the proxy's storage, saving gas and promoting proper contract behavior.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 18, 2024*	Public Report	Verichains Lab

Table 2. Report versions history