SECURITY AUDIT OF

KINGDOM RAIDS TOKEN AND VESTING SMART CONTRACTS

Public Report

Oct 03, 2022

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 prepared by Verichains Lab on Oct 03, 2022. We would like to thank the Kingdom Raids 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 Kingdom Raids Token and Vesting Smart 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 had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Kingdom Raids Token and Vesting Smart Contracts

Kingdom Raids is an RPG developed by a leading gaming studio, Alley Labs which has attracted more than 100 million downloads for their games. The game takes place in a fictional kingdom, "Dood Kingdom", a land of mystery and adventure.

Kingdom Raids Token is an ERC20 token that Kingdom Raids players can use in the game.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Kingdom Raids Token and Vesting Smart Contracts. It was conducted on commit `d227a5105143959855e51b61bbae76d78a6439dc` from git repository *https://github.com/kingdomraids/token-upgradeable*.

The last version of the proxy of Kingdom Raids token contract is deployed on Binance Smart Chain Mainnet at address `0x37b53894e7429f794B56F22a32E1695567Ee9913`.

The details of the proxy smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	TransparentUpgradeableProxy
Contract Address	0x37b53894e7429f794B56F22a32E1695567Ee9913
Compiler Version	v0.8.2+commit.661d1103
Optimization Enabled	Yes with 200 runs
Explorer	*https://bscscan.com/address/0x37b53894e7429f794B56F22a32E1695567Ee9913*

Table 1. The deployed smart contract details

The implementation contract is deployed on Binance Smart Chain Mainnet at address `0x589b8a8a614e959a6bde5b55be22df8956a04135`.

The details of the implementation contract are listed in Table 2.

FIELD	VALUE
Contract Name	KRS
Contract Address	0x589b8a8a614e959a6bde5b55be22df8956a04135
Compiler Version	v0.8.4+commit.c7e474f2
Optimization Enabled	No with 200 runs
Explorer	*https://bscscan.com/address/0x589b8a8a614e959a6bde5b55be22df8956a04135*

Table 2. The deployed smart contract details

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 was 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 3. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The Kingdom Raids Token and Vesting Smart Contracts was written in `Solidity` language, with the required version to be `^0.8.0`.

The Kingdom Raids uses `KRToken` contract to initial `Kingdom Raids Token` and release the tokens by the contracts correspond to categorize in tokenomics.

Table 3 lists some properties of the audited Kingdom Raids Token and Vesting Smart Contracts (as of the report writing time).

PROPERTY	VALUE
Name	Kingdom Raids Token
Symbol	KRS
Decimals	18
Total Supply	1,000,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 1,000,000,000 or 1 billion.

Table 4. The Kingdom Raids Token Contract properties

There are five contracts that the Kingdom Raids uses to release the tokens. They are `Advisor`, `EcosystemFund`, `Liquidity`, `Marketing` and `Team` contracts.

These contracts are `Vesting` contracts that release tokens every period after a cliffTime. Only the owner of the contract can claim the tokens.

Note: Sine the all contracts are upgradable, the contracts `owner` can upgrade them with the logic that is not in our audit scope.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of Kingdom Raids Token and Vesting Smart Contracts.

2.2.1. Market.sol - Misconfigure `eachReleaseAmount` value in Marketing contract MEDIUM

Following the source code, the number of tokens be released every month after the first unlock is:

```
(totalAllocation - firstUnlock) / 48 = (100e6- 100e6/ 100) / 48 = 2062500
```

But the `eachReleaseAmount` value is set with `2081250`. With the logic in the contract, the number of total released tokens is unchanged, but for lots of months, the number is larger than the average. Therefore, the tokenomic of the project can be broken.

UPDATES

- *Sep 29, 2022*: This issue has been acknowledged and fixed by the Kingdom Raids team in commit `bbc89a37ed2cba917a904502c1761f97f6140d87`. The formula was changed.

2.2.2. Upgradeable contract MEDIUM

The contracts in audit scope inherit `upgradeable` contracts which allow the deployer to change the logic. Any compromise to the `deployer` account may allow the hacker to take advantage of this.

RECOMMENDATION

We suggest changing all `upgradeable` abstract contract to normal contract.

UPDATES

- *Sep 29, 2022*: This issue has been acknowledged by the Kingdom Raids team.

APPENDIX

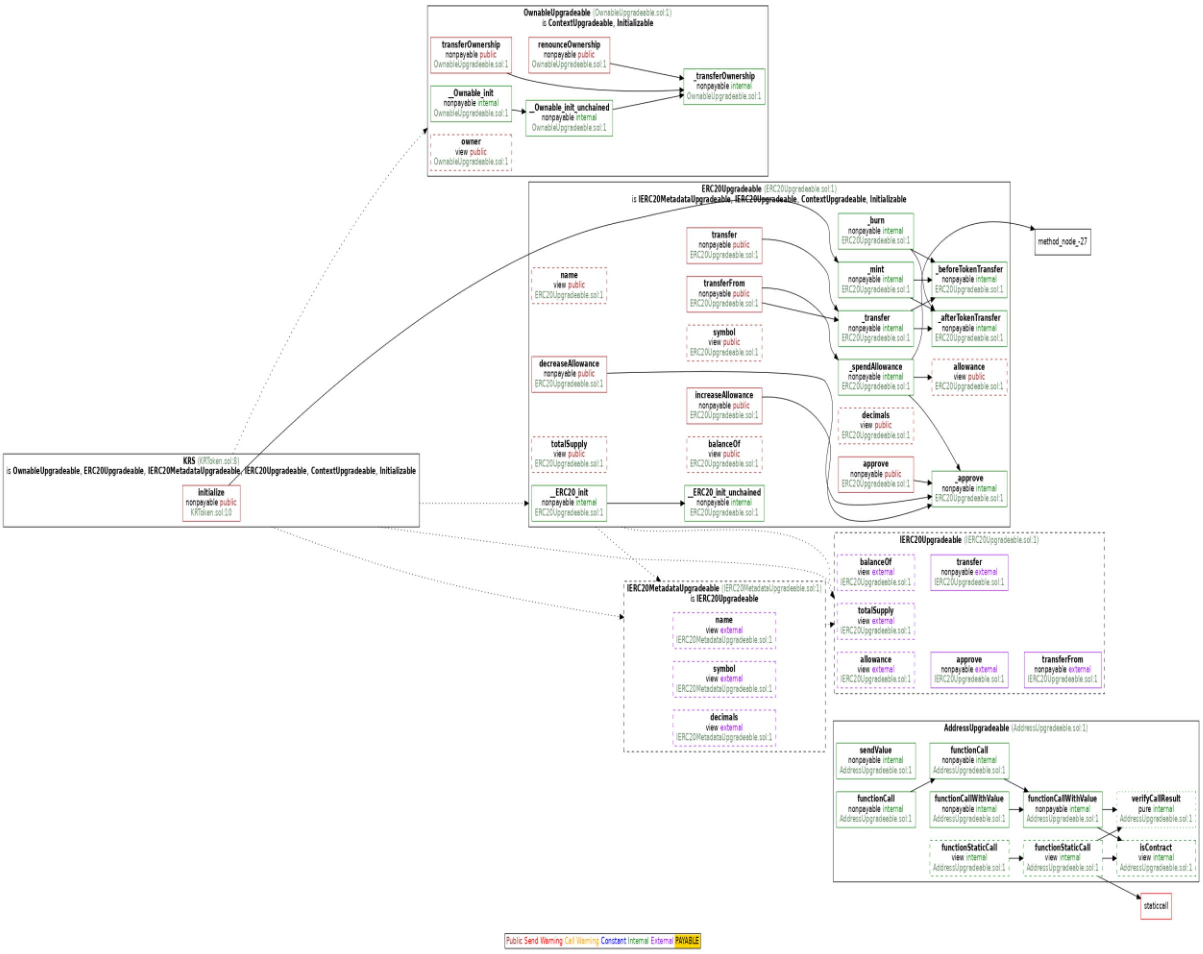

Image 1. Kingdom Raids Token call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Sep 29, 2022*	Public Report	Verichains Lab
1.1	*Oct 03, 2022*	Public Report	Verichains Lab

Table 5. Report versions history