SECURITY AUDIT OF

KRABOTS TOKEN SMART CONTRACT

Public Report

Jun 13, 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 Jun 13, 2022. We would like to thank the Krabots 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 Krabots Token Smart Contract. 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 no vulnerable issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Krabots Token Smart Contract

Krabots is a virtual robot combat game where anyone can earn tokens through creative physics-based battle.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Krabots Token Smart Contract.

It was conducted on commit 866c5ba9b5b39a719f95bb3e7343113f5e0e9f39 from git repository *https://github.com/krabotgame/krac-token/*.

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 1. 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 Krabots Token Smart Contract was written in Solidity language, with the required version to be 0.8.0.

Krabot token contract extends ERC20, ERC20Burnable, ERC20Snapshot, Pausable, Ownable contracts. With Ownable, by default, Contract Owner is also the contract deployer, but he can transfer ownership to another address at any time. Token Owner can pause/unpause contract using ERC20Pausable contract, user can only transfer tokens when the contract is not paused. The ERC20Burnable allows token holders to destroy both their own tokens and those that they have an allowance for. The ERC20Snapshot helps token Owner take a snapshot of the balances and total supply at a time for later access.

Table 2 lists some properties of the audited Krabots Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	Karbot
Symbol	KRAC
Decimals	18
Total Supply	100,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 100,000,000 or 100 million.

Table 2. The Krabots Token Smart Contract properties

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of Krabots Token Smart Contract.

2.3. Additional notes and recommendations

2.3.1. Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All safe math usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default, the only usage of safemath now is to have a custom revert message which isn't the case in the auditing contracts.

RECOMMENDATION

We suggest changing all methods from SafeMath library to normal arithmetic operator in the contract for readability and gas saving.

APPENDIX

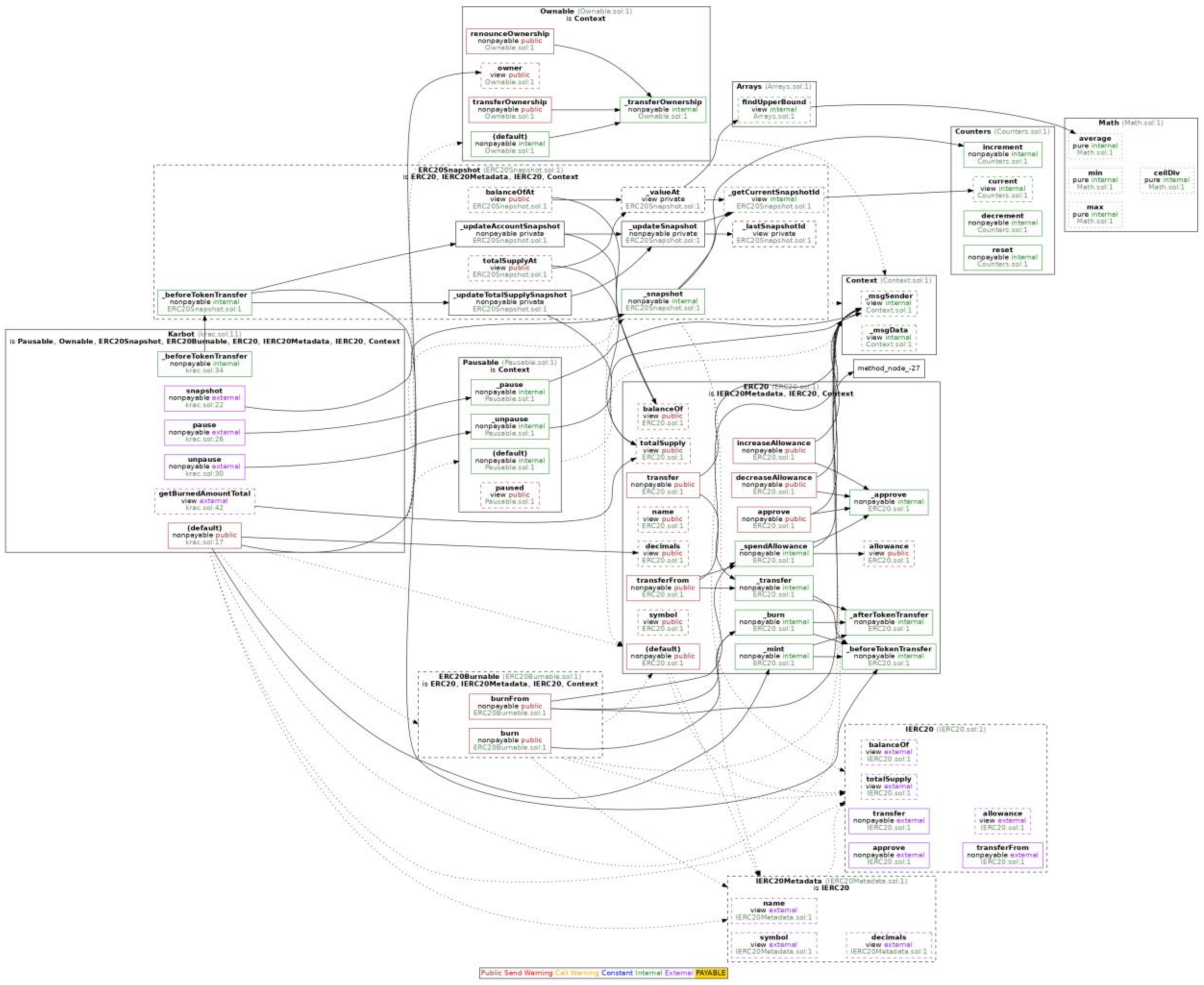

Image 1. Krabots Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jun 13,2022*	Public Report	Verichains Lab

Table 3. Report versions history