SECURITY AUDIT OF

CRYPTOGUARDS TOKEN SMART CONTRACT

Public Report

Dec 22, 2021

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 Dec 22, 2021. We would like to thank the CryptoGuards 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 CryptoGuards 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 issues in the application.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About CryptoGuards Token Smart Contract

With the success of CryptoCars and CryptoPlanes, continuing the CryptoCity Metaverse story series, the team continues to work together to release a new project - CryptoGuards - protect CryptoCity Metaverse against the Mysterious Forces from outside the Earth.

1.2. Audit scope

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

The audited contract is the CryptoGuards Token Smart Contract that deployed on Binance Smart Chain Mainnet at address 0xf81c707a9594f5876cf013cf8ecca9184e3d3250 (behind proxy at address 0x432c7cf1de2b97a013f1130f199ed9d1363215ba). The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	CGAR
Contract Implement Address	0xf81c707a9594f5876cf013cf8ecca9184e3d3250
Contract Proxy Address	0x432c7cf1de2b97a013f1130f199ed9d1363215ba
Compiler Version	v0.8.4+commit.c7e474f2
Optimization Enabled	Yes with 1000 runs
Explorer	*https://bscscan.com/address/0xf81c707a9594f5876cf013cf8ecca9184e3d3250*

Table 1. 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 2. 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

This table lists some properties of the audited CryptoGuards Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	CryptoGuards
Symbol	CGAR
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 3. The CryptoGuards Token Smart Contract properties

2.2. Contract codes

The CryptoGuards Token Smart Contract was written in Solidity language, with the required version to be 0.8.2.

The source codes consist of three contracts, five abstract contracts and two interfaces. Almost all source codes in the CryptoGuards Token Smart Contract imported OpenZeppelin contracts.

2.2.1. Initializable abstract contract

This is a base contract to aid in writing upgradeable contracts, or any kind of contract that will be deployed behind a proxy. The source code is referenced from OpenZeppelin's implementation.

2.2.2. ContextUpgradeable abstract contract

Provides information about the current execution context, including the sender of the transaction and its data. The source code is referenced from OpenZeppelin's implementation.

2.2.3. IERC20MetadataUpgradeable interface

Interface for the optional metadata functions from the ERC20 standard. The source code is referenced from OpenZeppelin's implementation.

2.2.4. IERC20Upgradeable interface

Interface of the ERC20 standard as defined in the EIP. The source code is referenced from OpenZeppelin’s implementation.

2.2.5. PausableUpgradeable abstract contract

Contract module allows children to implement an emergency stop mechanism that can be triggered by an authorized account. The source code is referenced from OpenZeppelin’s implementation.

2.2.6. OwnableUpgradeable abstract contract

Contract module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. The source code is referenced from OpenZeppelin’s implementation.

2.2.7. Ownable abstract contract

Contract module which provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. The source code is referenced from OpenZeppelin’s implementation.

2.2.8. ERC20Upgradeable contract

This is the contract implement ERC20 token with upgradable ability, adding transfer fee feature.

2.2.9. CGAR contract

This is the main contract in the CryptoGuards Token Smart Contract, which is inherited from ERC20Upgradeable contract.

2.2.10. Limiter contract

This contract implement buy/sell limit for CGAR contract.

2.3. Findings

The CryptoGuards Token Smart Contract was written in Solidity language, with the required version to be ^0.8.2. The source code was written based on OpenZeppelin's library.

During the audit process, the audit team had identified no vulnerable issues in the application.

APPENDIX

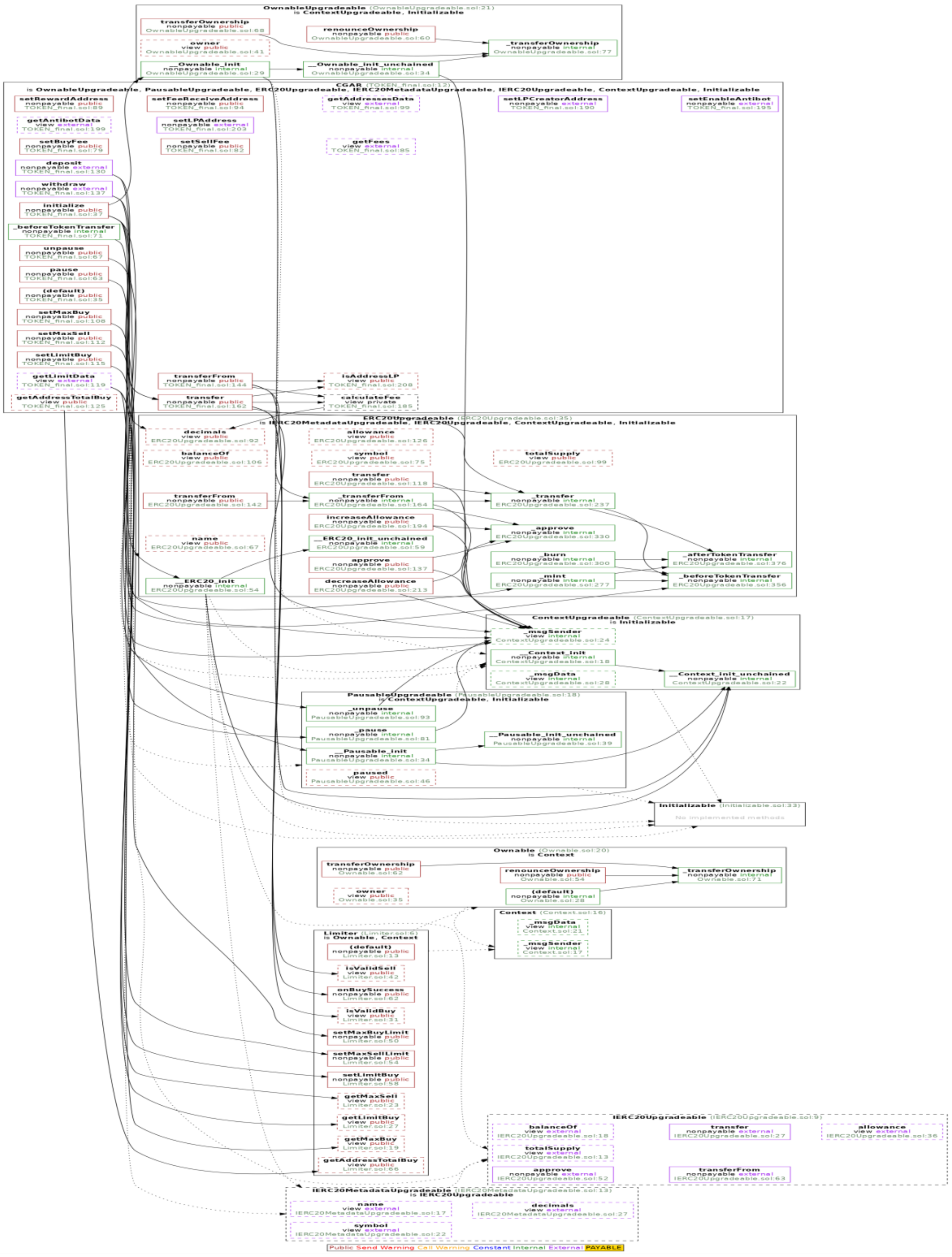

Image 1. CryptoGuards Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 22, 2021*	Public Report	Verichains Lab

Table 4. Report versions history