verichains

SECURITY AUDIT OF

CREATURE HUNTERS TOKEN SMART CONTRACT

Public Report

Dec 14, 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 14, 2021. We would like to thank the Creature Hunters 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 Creature Hunters 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 smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Creature Hunters Token Smart Contract

Creature Hunters is an online puzzle action game with blockchain technology based on the original animation produced by Mogi Hitosi and Chstudio, based in Seoul, Korea. Build the best team of yours, battle others, and easily earn money with your NFTsPlay for free, now!

1.2. Audit scope

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

It was conducted on commit 50919710d0905e44f445b193195fb6fe35030ab3 from git repository *https://github.com/Creaturehunters-CHTS/CHTS-contracts*.

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

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

PROPERTY	VALUE
Name	Creature Hunters Token S
Symbol	CHTS
Decimals	18
Total Supply	500,000,000 ($x10^{18}$) Note: the number of decimals is 18, so the total representation token will be 500,000,000 or 500 million.

Table 2. The Creature Hunters Token Smart Contract properties

The Creature Hunters Token Smart Contract was written in Solidity language, with the required version to be ^0.8.0. Almost all source codes in the Creature Hunters Token Smart Contract are imported from OpenZeppelin contracts.

Creature Hunters Token Smart Contract extends BEP20, BEP20Burnable, Pausable and Ownable contracts. by default, Token Owner is contract deployer, but he can transfer ownership to another address at any time. He can pause/unpause contract using Pausable contract, users can only transfer tokens when contract is not paused. The contract also add lock feature, the owner can lock/unlock any amount of tokens for any address, locked amount can not be transferred.

2.2. Findings

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

2.3. Additional notes and recommendations

2.3.1. Redundant lockedOf INFORMATIVE

lockedOf variable is set in lockAddress and unlockAddress but it is not used (calculate, require...) anywhere else and it can also be computed from lockedOfamount > 0 so it is redundant and waste of gas.

```
mapping(address => bool) public lockedOf;

function lockAddress(address account, uint256 amount) public onlyOwner re…
  turns (bool) {
    lockedOfamount[account] = amount;
    lockedOf[account] = true;
    return true;
}

function unlockAddress(address account) public onlyOwner returns (bool) {
    lockedOf[account] = false;
    lockedOfamount[account] = 0;
    return true;
}
```

RECOMMENDATION

Consider remove lockedOf variable.

APPENDIX

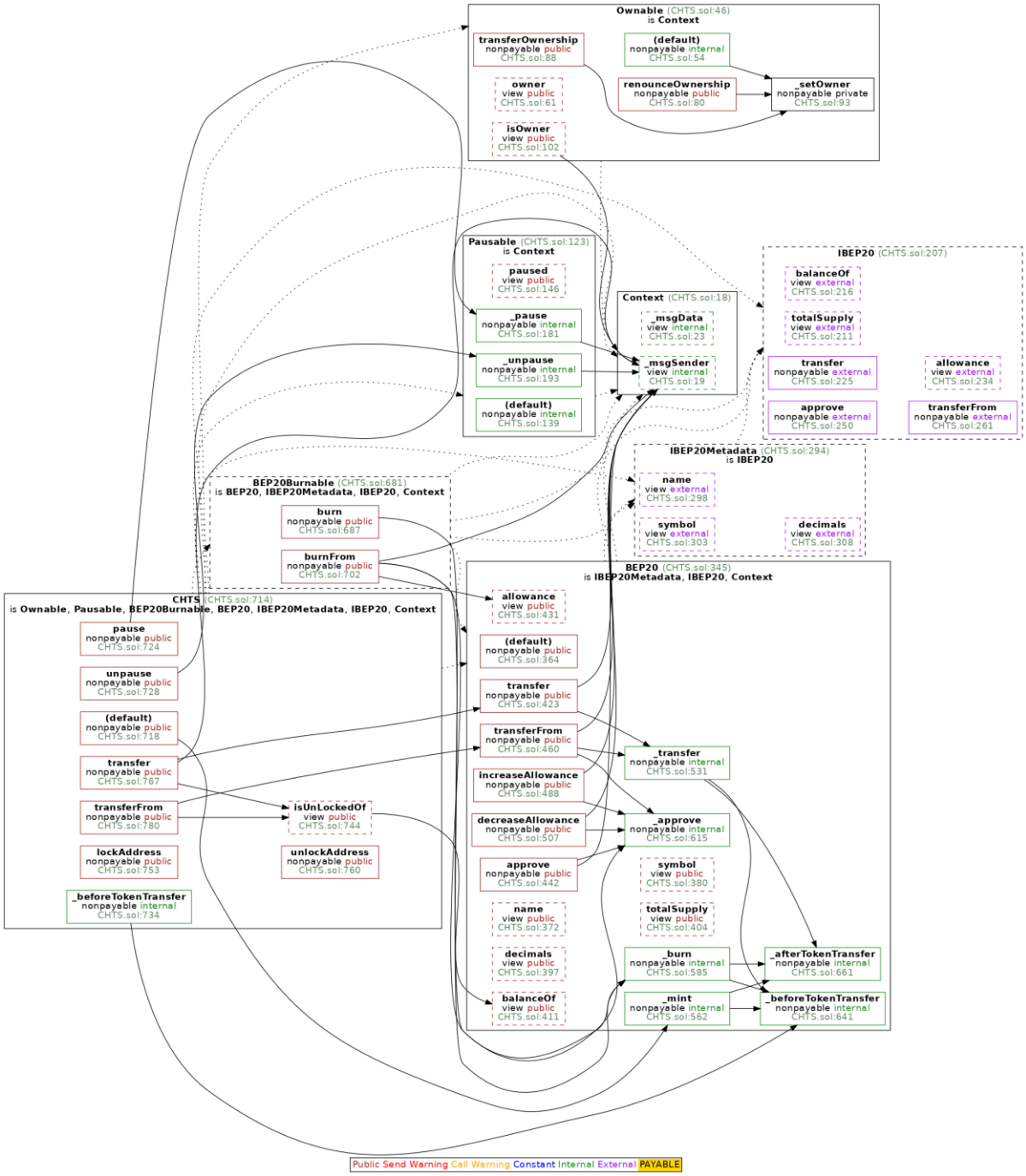

Image 1. Creature Hunters Token Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 14, 2021*	Public Report	Verichains Lab

Table 3. Report versions history