verichains

SECURITY AUDIT OF

TOKEN AND TOKEN VESTING SMART CONTRACTS

Public Report

Sep 08, 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 Sep 08, 2022. We would like to thank the CROS Games 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 Token and Token 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 **no vulnerable** issue in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Token and Token Vesting Smart Contracts

Ookeenga (OKG) is a 3D NFT gaming project which combines blockchain technology with breath-taking graphics, appealing world-building, and addictive gameplay to create a uniquely immersive play-to-earn experience developed by CROS Gamestudio and published by SPORES Network.

OKG is set in a world where insects have evolved and built a massive civilization in an ancient forest called Glaik (The Sacred Forest), which is divided into 2 factions: the Akhah (Pureblood) and the Ahika (Mixedblood).

In this chaotic fantasy world, players will be able to build forces, develop their own tribes, and engage in fierce battles to hold dominion over the Forest.

$OKG: is limited governance-supplied token. Players will need $OKG to buy/sell NFTs, breeding, or enhancing items. In addition, owning $OKG is equivalent to the player's investment in the OKG world.

1.2. Audit scope

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

There are two contracts in our audit contract. They are `OKGToken` and `PeriodicVesting` contracts

The `OKGToken` contract was deployed on Binance Smart Chain Mainnet at address `0x7758a52c1Bb823d02878B67aD87B6bA37a0CDbF5`. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	OKGToken
Contract Address	0x7758a52c1Bb823d02878B67aD87B6bA37a0CDbF5
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization Enabled	No with 200 runs

FIELD	VALUE
Explorer	*https://bscscan.com/address/0x7758a52c1Bb823d02878B67aD87B6bA37a0CDbF5*

Table 1. The deployed smart contract details

The `PeriodicVesting` contract was deployed on Binance Smart Chain Mainnet at address `0x8ebeD852909991Ef28A1AC9d05Bf033Bd9ba05c7`. The details of the deployed smart contract are listed in Table 2.

FIELD	VALUE
Contract Name	PeriodicVesting
Contract Address	0x8ebeD852909991Ef28A1AC9d05Bf033Bd9ba05c7
Compiler Version	v0.8.13+commit.abaa5c0e
Optimization Enabled	No with 200 runs
Explorer	*https://bscscan.com/address/0x8ebeD852909991Ef28A1AC9d05Bf033Bd9ba05c7*

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

2.1.1. OKGToken contract

`OKGToken` is an ERC20 token contract that extends `Ownable`, `Pausable` and `Ownable` contracts. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time. `ERC20Burnable` allows token holders to destroy both their own tokens and those that they have an allowance for.

Token Owner can pause/unpause contract using `Pausable` contract, user can only transfer unlocked tokens and only when the contract is not paused.

The contract also supports bot protection by `BPContract`, which will make a limit on buy/sell, add/remove blacklist, whitelist addresses and prevent sandwich attack. The owner can enable/disable bot protection at any time.

Note: Token Owner was transferred to a multi-sig contract on the mainnet.

2.1.2. PeriodicVesting contract

PeriodicVesting contract is a vesting contract that locks OKG tokens and releases tokens based on the timeline set by the owner. Almost all abstract contracts, contracts were inherited from OpenZeppelin contract that includes `Ownable` and `ReentrancyGuard`.

The contract implements the `withdraw` function which allows the owner to withdraw all vesting tokens in it.

2.2. Findings

During the audit process, the audit team found **no vulnerability** in the given version of the Token and Token Vesting Smart Contracts.

2.2.1. OKGToken - BPContract function INFORMATIVE

Since we do not control the logic of the `BPContract`, there is no guarantee that `BPContract` will not contain any security related issues. With the current context, in case the `BPContract` is compromised, there is not yet a way to exploit the Token and Token Vesting Smart Contracts, but we still note that here as a warning for avoiding any related issue in the future.

By the way, if having any issue, the `BPContract` function can be easily disabled anytime by the contract `owner` using the `setBpEnabled` function. In addition, `BPContract` is only used for a short time in token public sale IDO then the contract `owner` will disable it forever by the `setBotProtectionDisableForever` function.

2.2.2. Periodic - Token owner can mint unlimited amount of tokens INFORMATIVE

The token contract contains `mint` function for `onlyOwner` so the owner of the contract can `mint` unlimited amount of tokens without any cap.

UPDATES

- *Sep 08,2022*: This issue has been acknowledged and fixed by the CROS Games team, the `mint` function was removed from the source code.

2.2.3. PeriodicVesting - Redundant code in `addBeneficiary` function INFORMATIVE

The `addBeneficiary` contains a require statement verifying the `_beneficary` which was handled in the `zeroAddr` modifier.

```
function addBeneficiary(
    address _beneficiary,
    uint256 _totalAmt,
    uint256 _policy
) public onlyOwner isNotLocked zeroAddr(_beneficiary) {
require(
        _beneficiary != address(0),
        "beneficiary must not be address zero"
    );
    require(_policy < policies.length, "Policy not existed");
    require(beneficiaries[_beneficiary].allocated == 0, "Already added");
    beneficiaries[_beneficiary] = Beneficiary(_totalAmt, 0, _policy);

    beneficiariesList.push(_beneficiary);
}
modifier zeroAddr(address _addr) {
    require(_addr != address(0), "Address must not be zero");
    _;
}
```

RECOMMENDATION

- Remove redundant code for gas-saving.

UPDATES

- *Aug 24,2022*: This issue has been acknowledged and fixed by the CROS Games team.

2.2.4. PeriodicVesting - The `release` function may not work if the beneficiariesList is too large INFORMATIVE

The `release` function contains a for-loop fetching through `beneficiariesList` which consumes gas following the length of `beneficiariesList`. So, if the length of `beneficiariesList` is too large, the `release` transaction will run out of gas.

```
function release() external override nonReentrant onlyOwner {
        for (uint256 i = 0; i < beneficiariesList.length; i++) {
            claim(beneficiariesList[i]);
```


verichains

```
        }
    }
```

APPENDIX

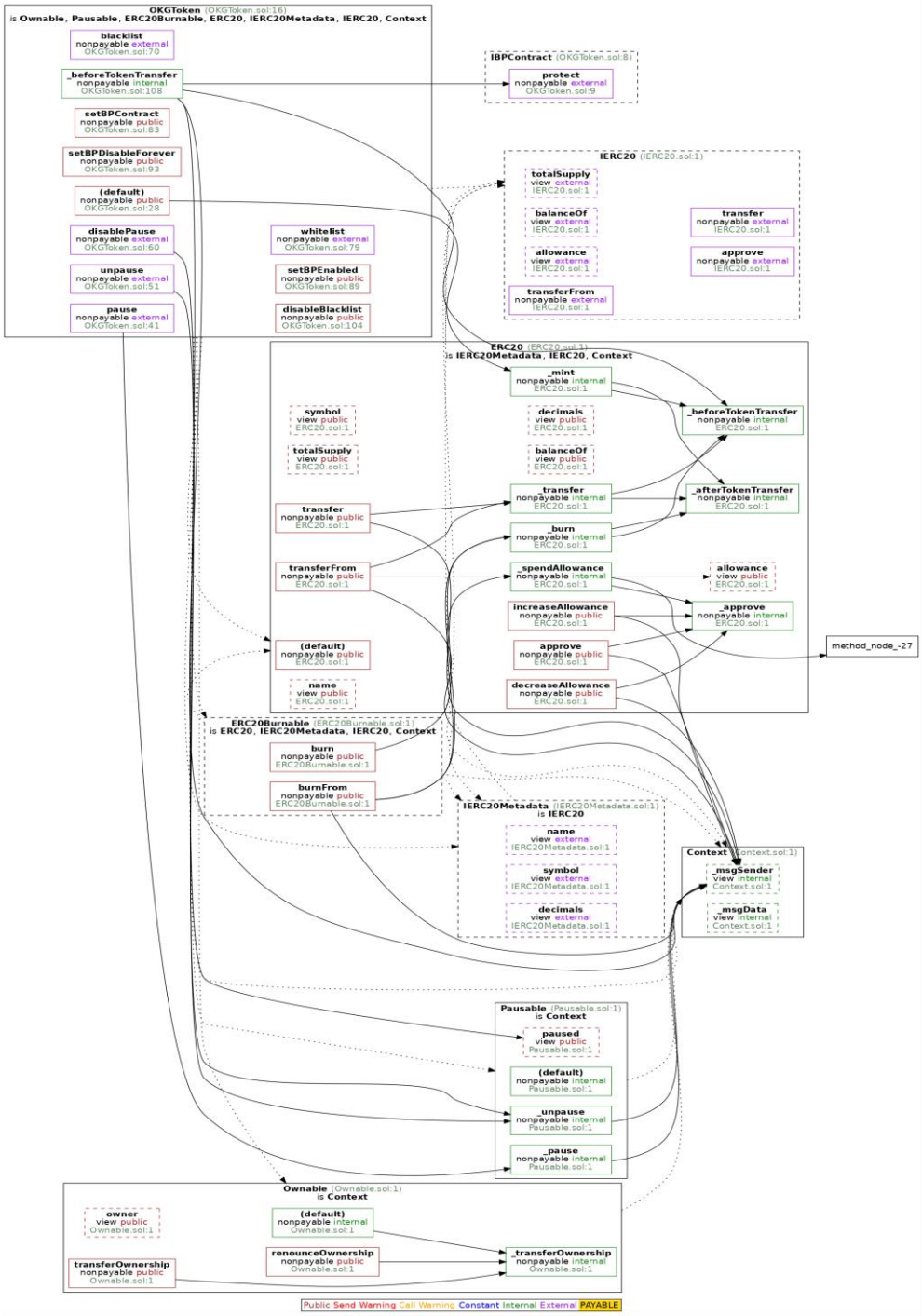

Image 1. OKGToken contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Aug 24, 2022*	Public Report	Verichains Lab
1.1	*Sep 08, 2022*	Public Report	Verichains Lab

Table 4. Report versions history