SECURITY AUDIT OF

METAGEAR TOKEN SMART CONTRACTS

Public Report

Jan 19, 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 Jan 19, 2022. We would like to thank the MetaGear 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 MetaGear Token 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 MetaGear Token Smart Contracts

MetaGear is a game that shows creativity in assembling robots to fight. Challenge yourself with a wide variety of game modes.

MetaGearToken is an ERC20 token that MetaGear players can use in the game.

1.2. Audit scope

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

The audited contract is the MetaGear Token Smart Contracts that deployed on Binance Smart Chain Mainnet at address 0xb4404DaB7C0eC48b428Cf37DeC7fb628bcC41B36. The details of the deployed smart contract are listed in Table 1.

FIELD	VALUE
Contract Name	MetaGearToken
Contract Address	0xb4404DaB7C0eC48b428Cf37DeC7fb628bcC41B36
Compiler Version	v0.8.5+commit.a4f2e591
Optimization Enabled	Yes with 999999 runs
Explorer	*https://bscscan.com/address/0xb4404DaB7C0eC48b428Cf37DeC7fb628bcC41B36*

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

The initial review was conducted in Jan 14, 2021 and a total effort of 3 working days was dedicated to identifying and documenting security issues in the code base of the MetaGear Token.

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

PROPERTY	VALUE
Name	MetaGear Token
Symbol	GEAR
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 3. The MetaGear Token Smart Contracts properties

2.2. Contract codes

The MetaGear Token Smart Contracts was written in Solidity language, with the required version to be 0.8.5.

MetaGear token contract extends ERC20Snapshot, Pausable, AccessControl, UsingLiquidityProtectionService contracts.

The PAUSER_ROLE can pause/unpause contract using Pausable contract, user can only transfer unlocked tokens and only when the contract is not paused. The ADMIN_ROLE also creates a record of the balances and total supply at any time through the snapshot function.

The contract has a mechanism to block accounts through a bunch of Blacklist functions. They support the MetaGear team to reduce the impact of suspect accounts. In addition, the contract implements the burnFrom function. So an allowance account can call this function to remove the owner balances.

For protecting liquidity pool when the first time adding liquidity, the MetaGear token contract intergrates with a bot prevention mechanism by extends

UsingLiquidityProtectionService and using LiquidityProtection_beforeTokenTransfer function at 0x0649601Ca10D68792AA0285bD25E1E5938FED8E5 to check.

2.3. Findings

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

2.4. Additional notes and recommendations

2.4.1. UsingLiquidityProtectionService.sol - LiquidityProtection_beforeTokenTransfer function INFORMATIVE

Since we do not control the logic of the LiquidityProtection_beforeTokenTransfer at 0x0649601Ca10D68792AA0285bD25E1E5938FED8E5 address, there is no guarantee that LiquidityProtection_beforeTokenTransfer will not contain any security related issues. With the current context, in case the LiquidityProtection_beforeTokenTransfer is compromised, there is not yet a way to exploit the MetaGear Token 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 LiquidityProtection_beforeTokenTransfer function can be easily disabled anytime by the contract ADMIN_ROLE using the disableProtection function.

2.4.2. MetaGearToken.sol - 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.

UPDATES

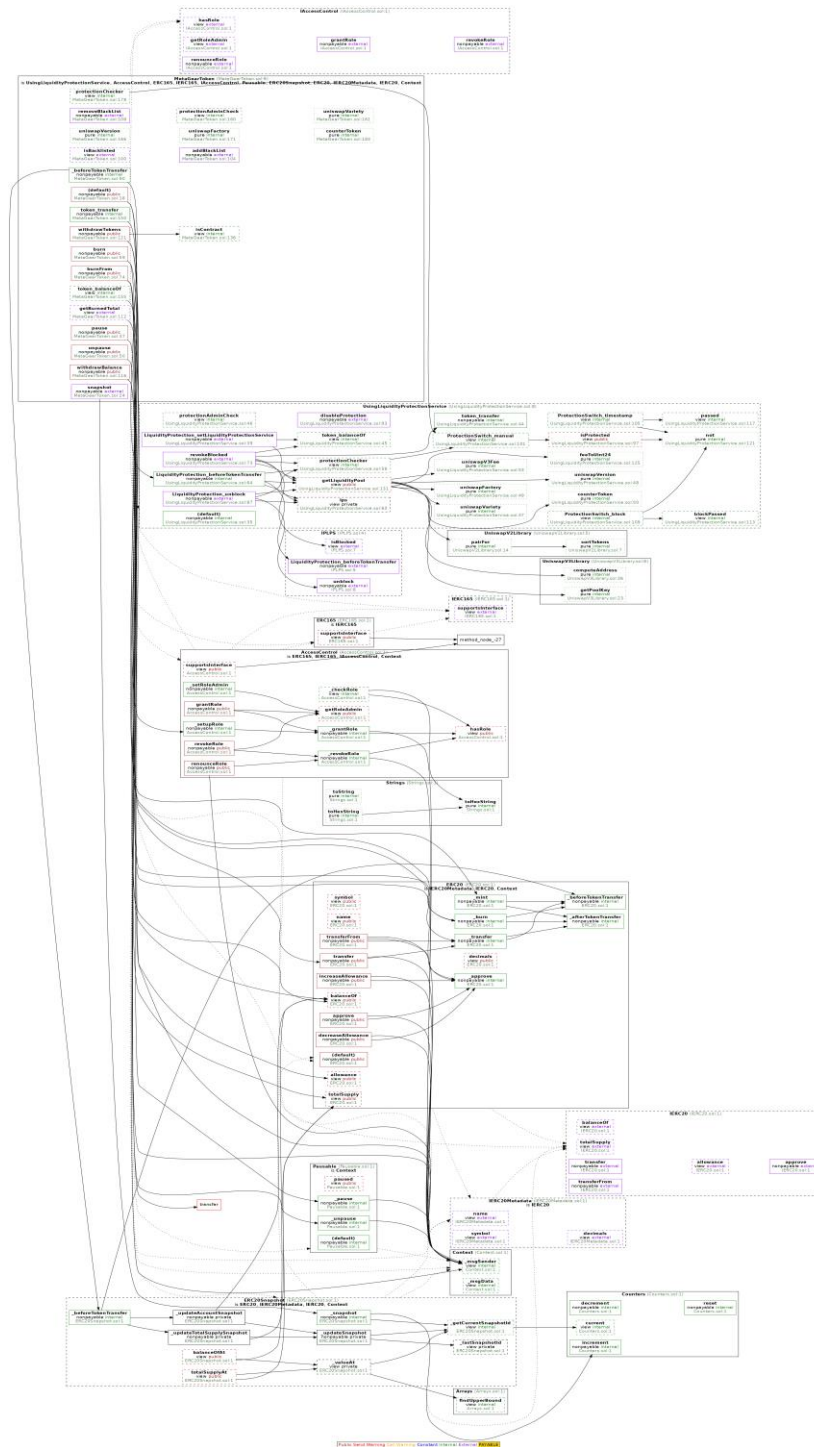- *Jan 17, 2022*: This recommendation has been acknowledged by the MetaGear team.

APPENDIX

Image 1. MetaGear Token Smart Contracts call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 17,2022*	Public Report	Verichains Lab
1.1	*Jan 19,2022*	Public Report	Verichains Lab

Table 4. Report versions history