SECURITY AUDIT OF

SECONDTOKEN SMART CONTRACTS

Public Report

Feb 07, 2024

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 was prepared by Verichains Lab on Feb 07, 2024. We would like to thank the DOS Labs 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 SECONDToken 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 contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About SECONDToken Smart Contracts

SECOND token is a ERC20 token with additional functionalities. This token is a implementation of The Omnichain Fungible Token (OFT) standard. The OFT standard allows fungible tokens to be transferred across multiple blockchains without asset wrapping, middlechains, or liquidity pools.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the SECONDToken Smart Contracts. It was conducted on commit `0e90c29675374c9af2f5a94a26e415a50d2e2f60` from git repository link: *https://github.com/DOSLabs/Smart-Contracts/tree/main/auditsV4*.

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 were 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

DOS Labs acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. DOS Labs understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, DOS Labs agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the DOS Labs will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the DOS Labs, the final report will be considered fully accepted by the DOS Labs without the signature.

2. AUDIT RESULT

2.1. Overview

The SECONDToken Smart Contracts was written in `Solidity` language, with the required version to be `^0.8.20`. The source code was written based on OpenZeppelin library and Layerzerolabs library.

The `SECONDtoken` is an ERC20 token distinguished by its supplementary functionalities. It represents an implementation of The Omnichain Fungible Token (OFT) standard, which facilitates the transfer of fungible tokens across multiple blockchains without the need for asset wrapping, middlechains, or liquidity pools.

The mechanism of this standard involves burning tokens on the source chain whenever an omnichain transfer is initiated. Subsequently, a message is dispatched via the protocol, triggering a function call to the destination contract. This call is responsible for minting an equivalent number of tokens as those burned, effectively establishing a unified supply across both networks.

The Omnichain Fungible Token (OFT) utilizes a dedicated endpoint to manage the minting of tokens on the destination chain. This endpoint is embodied as a contract deployed specifically on the destination chain, entrusted with the responsibility of minting tokens upon receiving a message from the source chain. The endpoint contract is characterized by a straightforward structure, featuring a single function named mint(). When invoked by the protocol in response to a message from the source chain, the mint() function executes the task of minting an equivalent number of tokens as those burned on the source chain. It's crucial to note that the endpoint is established during the deployment of the OFT contract and remains immutable thereafter.

During the deployment of the contract, an admin is a designated address that possesses a predefined supply of tokens.

2.2. Findings

During the audit process, the audit team had identified no vulnerable issue in the contract code.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Feb 07, 2024*	Public Report	Verichains Lab

Table 2. Report versions history