verichains

SECURITY AUDIT OF

HOLDSTATION TOKEN

Public Report

Oct 24, 2023

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 Oct 24, 2023. We would like to thank the Holdstation 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 Holdstation Token. 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 Holdstation Token

A governance token that enables holders to participate in the decision-making process through voting on DAO proposals, #realyield farming and multiple fees discount.

HOLD token distribution follows a fixed supply, decaying emission model as a general principle. This means that as time passes, the token emission decreases according to a fixed schedule. This distribution model is designed to reward users who stake or hold tokens for longer periods.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the smart contracts of Holdstation Token. It was conducted on commit `1654ebeb9c309c2365c3e7293bc28353cc5d7ba6` from git repository *https://gitlab.com/hspublic/hs-launchpad*.

The latest version of the following file was made available in the course of the review:

SHA256 Sum	File
`661a9cc445126848721c9178f7f2e0fa988234f5f2f282a233da6bb6df1a87db`	`token/HoldToken.sol`

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

Holdstation 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. Holdstation understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Holdstation 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.

2. AUDIT RESULT

2.1. Overview

The Holdstation Token was written in `Solidity` language, with the required version to be `^0.8.10`. The source code was written based on OpenZeppelin's libraries: `Ownable`, `ERC20Capped`.

The contract has a function called claimStuckTokens() function that enables the owner to recover tokens. This function works for both tokens and Ether (do not for HOLD token).

The smart contract is `ERC20` implementation that have some properties (as of the report writing time):

PROPERTY	VALUE
Name	Holdstation
Symbol	HOLD
Decimals	18
Total Supply	30,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 30 million.

Table 2. The Holdstation Token properties

For the ERC20 token, the security audit team has the list of centralization issues below:

Checklist	Status	Passed
Fee modifiable	No	Yes
Mintable	No	Yes
Pausable	No	Yes
Trading cooldown	No	Yes
Has blacklist	No	Yes
Has whitelist	No	Yes

Table 3. The decentralization checklist

2.2. Findings

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

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Oct 24, 2023*	Public Report	Verichains Lab

Table 4. Report versions history