verichains

SECURITY AUDIT OF

STAKING SMART CONTRACT

Public Report

Mar 05, 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 Mar 05, 2024. We would like to thank the Hydrogen 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 Staking 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 Staking Smart Contract

The Staking Smart Contract is deployed on Blast Blockchain (an EVM-forked). It is a gem in Blast ecosystem.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the Staking Smart Contract. It was conducted on commit `86eb1c5da50e4db30564060bf3868a32ee8f9416` from git repository link *https://github.com/KelvinThai/Blast_LpStaking.git*.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
5551e9147d20ec133ca56f6a862c60cffd4600f7beb629951c5782a7ef0470bb	contracts/src/HsETH.sol
9c6022fca22d15928d2175d7c5d3d29032f144dc26d0c4902b9a4711bf9bd331	contracts/src/HsUSDB.sol
e3323e4283eb8c47f42460de0c668f2d60e5d1c3e3602f8705603fb231cf4350	contracts/src/Staking.sol
fcbe450fdac67ca1d45bfc2a1a7b460ea3f8454e13ef357bc88461c09c4f040b	contracts/src/UsdbStaking.sol
656702168fa4422710936bc8eeb37ae512f72d491985994c15f86d37543807ed	contracts/src/UsdbVault.sol
78098fc1e8dc2a35ac115fc3c50e357f8bfdb8afba2bcc96e84926e5dc580c74	contracts/src/UsdbVaultFactory.sol
686f20dcc4723b3d6287b8b56159fb43e41761d853b5fecbab11e0ae6f3f6d80	contracts/src/Vault.sol
804aa28b0bf604900fa7f1fc06f5f8edc7e3dd75a229c8b1809309eff16cefa2	contracts/src/VaultFactory.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

Hydrogen 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. Hydrogen understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, Hydrogen 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 Hydrogen will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the Hydrogen, the final report will be considered fully accepted by the Hydrogen without the signature.

2. AUDIT RESULT

2.1. Overview

- **VaultFactory.sol** and **UsdbVaultFactory.sol**: Create own user's vaults.
- **Vault.sol** and **UsdbVault.sol**: Add `ETH`/`USDB` as collateral and borrow `hsETH`/`hsUSDB` with 1:1 ratio.
- **Staking.sol** and **UsdbStaking.sol**: Stake or unstake `hsETH`/`hsUSDB` to earn rewards.

The properties of ERC20 tokens:

- **HsETH**

PROPERTY	VALUE
Name	hsETH
Symbol	hsETH
Decimals	18
Total Supply	unidentified

Table 2. The hsETH Smart Contract properties

- **hsUSDB**

PROPERTY	VALUE
Name	hsUSDB
Symbol	hsUSDB
Decimals	18
Total Supply	unidentified

Table 3. The hsUSDB Smart Contract properties

2.2. Findings

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

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Feb 23, 2024*	Public Report	Verichains Lab
1.1	*Mar 05, 2024*	Public Report	Verichains Lab

Table 4. Report versions history