verichains

SECURITY AUDIT OF

STARKFINANCE EXCHANGE

Public Report

Mar 14, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Starknet	Starknet is an Ethereum layer-2 scaling solution that uses a zero-knowledge rollup based on StarkWare Industry's trustless "STARK" proof.
Cairo	Cairo is A STARK-based Turing-complete language for writing provable programs on blockchain. Cairo enables developers to use proof technology.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
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 14, 2024. We would like to thank the StarkFinance 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 StarkFinance Exchange. 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 StarkFinance Exchange

StarkFinance Exchange is leading decentralized exchange on Starknet. Users can swap, earn, and provide liquidity with low fees and most efficient trading functions.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the StarkFinance Exchange that was deployed on Starknet.

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

Contract	Address
Factory	0x64613af31ea3b32fd7608f6fde869de79304b19c80c8b759704060d5b392046
Router	0x2a6396450ee16b429936b7331d19c0a63b54d0712bee7601c70716ee8d8eb77

It was conducted on commit `ecac048ecdd7d0eefe3bcf6534dd4d41f4a79709` from git repository link *https://github.com/starkfinance-organization/starkfinance-exchange-contracts.git* that forked from *StarkDefi* repository (based on the Uniswap V2). The security audit team approaches to the code review were based on different between 2 repositories.

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

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

2. AUDIT RESULT

2.1. Overview

The StarkFinance Exchange was written in the `Cairo` language. The contract makes use of `OpenZeppelin` libraries like `Proxy` and `ERC20`.

2.1.1. Router

The `StarkFinanceRouter` is a smart contract written in Cairo language for StarkNet, which was based from Uniswap V2. It facilitates token swapping and liquidity management, following the principles of Uniswap V2 Router. Users can perform various operations like adding/removing liquidity and token swapping efficiently. The contract provides interfaces for ERC20 tokens, stores factory addresses, and ensures error-free operations.

2.1.2. Factory

This Cairo smart contract, `StarkFinanceFactory`, facilitates the creation and registration of new pairs for decentralized exchanges on StarkNet. It handles token sorting, fee management, and emits events upon pair creation.

2.2. Findings

During the audit process, the audit team found no vulnerability in the given version of StarkFinance Exchange.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Mar 14, 2024*	Public Report	Verichains Lab

Table 2. Report versions history