verichains

SECURITY AUDIT OF

COLEND ORACLE SMART CONTRACT

Public Report

Jul 31, 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 Jul 31, 2024. We would like to thank the Colend 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 Colend Oracle 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 Colend Oracle Smart Contract

The Oracle of Colend protocol utilizes a price oracle to fetch asset prices, crucial for lending operations. It relies on Pyth, an off-chain oracle, to enhance security and data accuracy by aggregating prices from trusted sources. The protocol employs a pull model over a push model for additional security benefits, such as reduced attack surfaces, control over data retrieval, and enhanced validation. This approach ensures up-to-date and reliable price data, minimizing risks of price manipulation.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of Colend Oracle Smart Contract.

It was conducted on commit `b695fa3ed9bf7c68b7454c9ce9e8b41aa8217a38` from the GitHub repository *tobyColend/aave-v3-core-verichain*, specifically on the `feat/stcore-aggregator` branch.

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

SHA256 Sum	File
b7f367fce77bb63b9f9db6fae664a3cb547ca1b1ad1587ad8e6d285c60f3d720	contracts/misc/AaveOracle.sol
1cbbc53ac48cbcc7fa4628a212ddd039c212222e548a77c3997ab99a462e2c69	contracts/misc/FallbackOracle.sol
a13c95dde173156f042a7f156e5563a2513185563c67ab74c13e3d53dfab0419	contracts/misc/SupraFallbackOracle.sol
62a418d73e9aadf83911181b845168dcb2484ea4d3323ab8e952a7ee99e341d5	contracts/misc/StCOREAggregatorV3.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

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

2. AUDIT RESULT

2.1. Overview

The Colend Oracle Smart Contract was written in `Solidity` language, with the version is `0.8.10`.

2.1.1. AaveOracle.sol

The `AaveOracle` smart contract is used to obtain asset prices, manage price sources, and update the fallback oracle. It prioritizes **Pyth** Price Feeds but resorts to a fallback oracle if Pyth returns a price ≤ 0. The contract is governed by **Aave** governance. It includes a constructor to initialize the oracle with addresses and price feed IDs, and functions to set asset price feed IDs and fallback oracle. It ensures that only asset listing or pool admins can perform certain actions. The contract also provides methods to get individual and multiple asset prices.

2.1.2. FallbackOracle.sol

The `FallbackOracle` contract in Solidity retrieves asset prices, using **Chainlink** aggregators primarily and a fallback oracle as a secondary source. It ensures price data is current by implementing a staleness threshold. Only asset listing or pool admins can set asset sources and update the fallback oracle. The contract is initialized with the pool addresses provider, main oracle, staleness threshold, and asset sources. It includes functions for setting and retrieving asset sources, prices, and the staleness threshold, ensuring robust and reliable price data management.

2.1.3. SupraFallbackOracle.sol

The `SupraFallbackOracle` smart contract retrieves asset prices using a primary oracle (**SupraSValueFeed**) and a fallback oracle for redundancy. It maintains a mapping of asset pair indexes and ensures price data freshness by enforcing a staleness threshold. Only authorized admins can update settings such as staleness thresholds, asset pair indexes, and the fallback oracle. The contract includes functions to fetch individual or multiple asset prices and handles price data validation and consistency checks.

2.1.4. StCOREAggregatorV3.sol

The `StCOREAggregatorV3` smart contract provides a **Chainlink AggregatorV3** interface that utilizes **Pyth** network price feeds and integrates with the **StCore Earn** contract. It fetches and computes asset prices by combining the core price from **Pyth** and the current exchange rate from the Earn contract. The contract includes functions to retrieve price data, decimals, descriptions, and timestamps, and uses the price feed ID for accurate price representation without storing round ID information on-chain.

2.2. Findings

During the audit process, the audit team found no vulnerabilities in the given version of Colend Oracle Smart Contract.

2.3. Additional Notes

2.3.1. INFORMATIVE - Be careful with the mock contracts when deploying

All files in mock folder should not be used for deployment. They are only for testing purposes. In history, the AAVE team has a mistake in deploying `PriceOracle` contract in mock folder for fallback oracle and be exploited by an attacker (ref).

UPDATES

- ***Jul 31, 2024***: The note was acknowledged by the Colend Oracle Smart Contract team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jul 31, 2024*	Public Report	Verichains Lab

Table 2. Report versions history