verichains

SECURITY AUDIT OF

LIVETRADE SMART CONTRACTS

Public Report

Feb 08, 2022

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 prepared by Verichains Lab on Feb 08, 2022. We would like to thank the LiveTrade 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 LiveTrade 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 smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About LiveTrade Smart Contracts

LiveTrade LTD was established with the goal to become the connection hub of the world's financial markets rather than struggling to fight against thousands of competitors in the market. We offer a diversity of services, including funding, loans and financial and commercial assistance for small and medium-sized businesses that are hopeful of jumping into the immense international market.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the LiveTrade Smart Contracts.

It was conducted on commit 2a42c8f74f095db35850887a111deb3ae2b7306b from git repository *https://gitlab.com/it323/ltdsmartcontract*.

There are 2 files in our audit scope. They are LTDToken.sol and LTDTreasuryVester.sol files.

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 was 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

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

This audit focused on identifying security flaws in code and the design of the LiveTrade Smart Contracts.

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

FILE	SHA256 SUM
LTDToken.sol	6203b618f24d35545f2340f62c1eafba9926b86b4a3d821a1aa558f063ddf946
LTDTreasuryVester.sol	6bedfaa58fbb15ad49cc23a8dd61212dffde41ade85700580c6af5fa4c05005d

2.1.1. Contract codes

The LiveTrade Smart Contracts was written in Solidity language, with the required version to be ^0.8.2.

2.1.2. LTD token contract

This is the contract implement ERC20 token. The source code is referenced from OpenZeppelin's implementation.

LTD token extends ERC20, ERC20Snapshot, Owable and ERC20Pausable contracts. ERC20Snapshot help Token Owner take a snapshot of the balances and total supply at a time for later access. Token Owner can pause/unpause contract using ERC20Pausable contract, user can only transfer tokens when contract is not paused.

The contract has a mechanism to block accounts through a bunch of Blacklist functions. They support the LiveTrade team to reduce the impact of suspect accounts.

This table lists some properties of the audited LiveTrade Smart Contracts (as of the report writing time).

PROPERTY	VALUE
Name	LTD Token
Symbol	LTD

PROPERTY	VALUE
Decimals	18
Total Supply	500,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 500,000,000 or 500 million.

Table 2. The Live Trade Token Smart Contract properties

2.1.3. LTDTreasuryVester contract

LTDTreasuryVester contract is a vesting contract which releaes the token for investor following the time. For each this contract, it reposible for an investor. All information like ltdAmount, treasuryName, releaseTime were set by the owner in the contructor. The recipient was also set in this time, but it can be changed by the owner through setRecipient function.

2.2. Findings

During the audit process, the audit team found no vulnerability issues in the given version of LiveTrade Smart Contracts.

2.3. Additional notes and recommendations

2.3.1. LTDTreasuryVester.sol - Unsafe using transfer method through IERC20 interface INFORMATIVE

The claim function uses transfer method to call the function from the token contract. With the LTD token contract in the audit scope, it doesn't have any problems. But if the contract was used for vesting another token, it may cause issues in future development.

For instance with an insecurity contract, the transfer function can return false with the function call failure instead of returning true or revert like ERC20 Oppenzepplin. With claim logic, the user doesn't receive anything while the status value still changes.

```
77  function claim() public {
78          require(msg.sender == recipient, "TreasuryVester::Claim You a…
    re not recipent");
79          require(block.number >= nextBlockVesting, "TreasuryVester::Cl…
    aim: Not time yet");
80
81          IERC20(ltdToken).transfer(recipient, nextAmountVesting);
82          setNextVesting();
83      }
```

Snippet 1. LTDTreasuryVester.sol Unsafe using `transfer` method in `claim` function

RECOMMENDATION

We suggest using SafeERC20 library for IERC20 and changing all transfer, transferFrom method using in the contract to safeTransfer, safeTransferFrom which is declared in SafeERC20 library to ensure that there is no issue when transferring tokens.

UPDATES

- *Feb 08, 2022*: This issue has been acknowledged and fixed by the LiveTrade team in commit 2a42c8f74f095db35850887a111deb3ae2b7306b.

2.3.2. LTDTreasuryVester.sol - Unnecessary usage of SafeMath library in Solidity 0.8.0+ INFORMATIVE

All safe math usage in the contract are for overflow checking, solidity 0.8.0+ already do that by default. We suggest using normal operators for readability and gas saving.

RECOMMENDATION

We suggest changing all methods from SafeMath library to normal arithmetic operator in the contract and remove this library from the source code.

APPENDIX

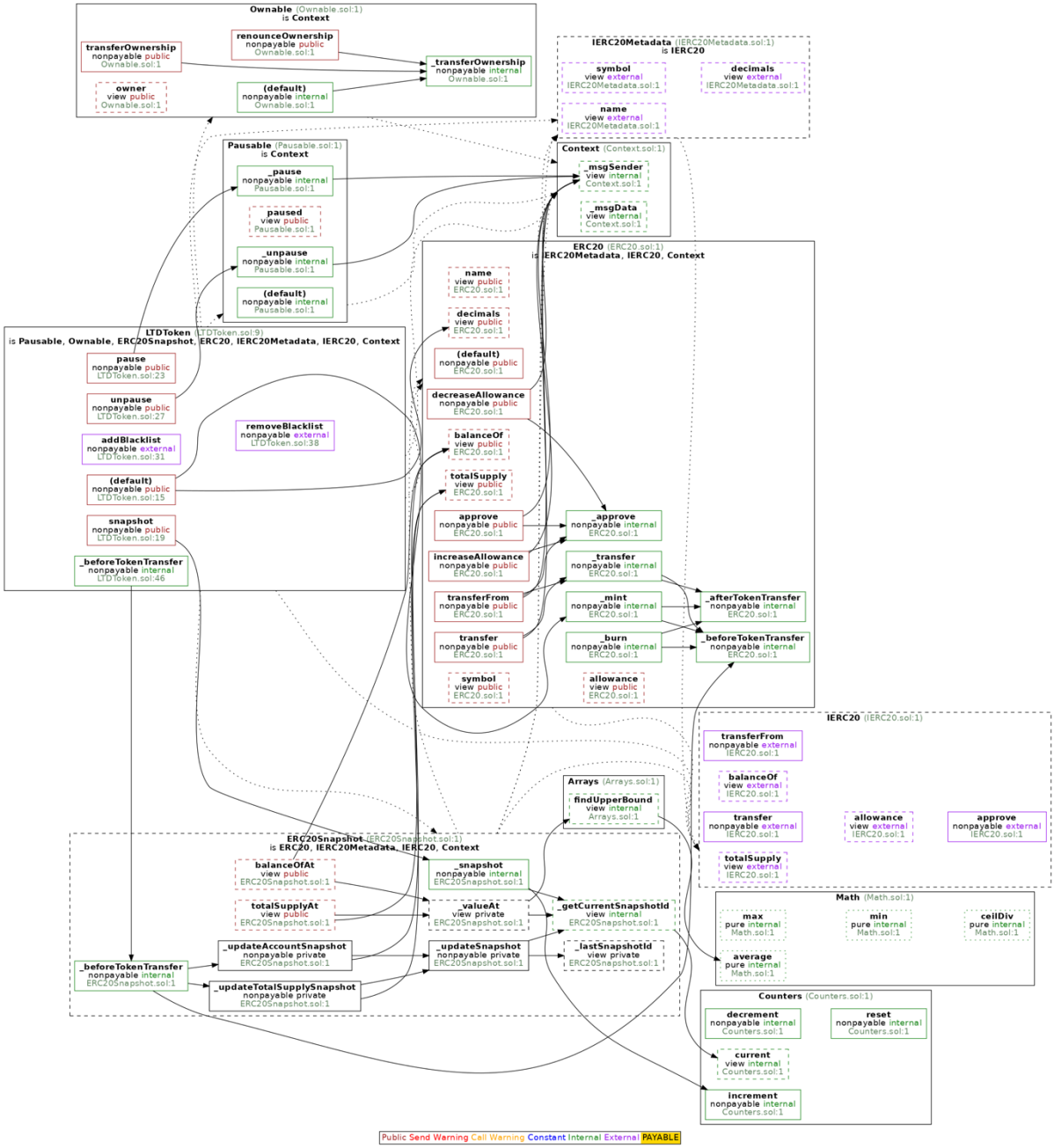

Image 1. LiveTrade Token Smart Contract call graph

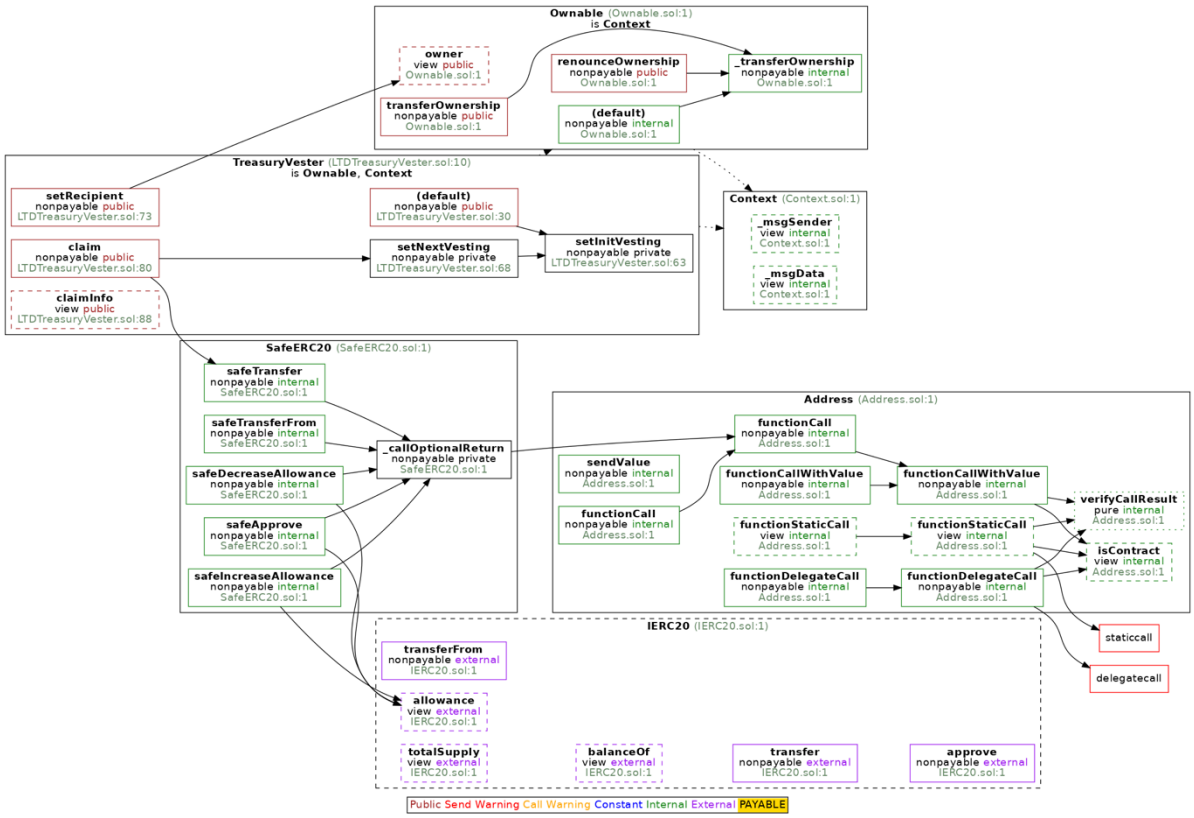

Image 2. LiveTrade TreasuryVester Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 26, 2022*	Public Report	Verichains Lab
1.1	*Feb 08, 2022*	Public Report	Verichains Lab

Table 3. Report versions history