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 some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY **5**

1.1. About LiveTrade Smart Contracts **5**

1.2. Audit scope **5**

1.3. Audit methodology **5**

1.4. Disclaimer **6**

2. AUDIT RESULT **8**

2.1. Overview **8**

2.1.1. LiveTradeMasterChef contract 8

2.1.2. BaseNFTBountyStake contract 8

2.1.3. LTDMigration contract 8

2.1.4. LiveTradeStaking contract 9

2.1.5. LTDStakingNFTWhitelist contract 9

2.1.6. LTDSwapStock contract 9

2.1.7. BaseDipoLTD contract 9

2.2. Findings **10**

2.2.1. LTDDipoContract.sol - claimTGE without checking for startTimeTGE CRITICAL 10

2.2.2. Comparing boolean with true/false INFORMATIVE 11

2.2.3. LTDDipoContract.sol - governanceRecoverUnsupported can withdraw any ERC20 Token INFORMATIVE 11

2.2.4. LTDDipoContract.sol - Unnecessary nonReentrant in setOwnerIDO INFORMATIVE 12

3. VERSION HISTORY **13**

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 5ad836338ce42b4b437e5827428c4161cd9f5b60 from git repository *https://gitlab.com/it323/ltdsmartcontract*.

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

SHA256 Sum	File
67486c1f7e407967edaa5fb509d23a64c3c02f85c971138517c567ad7acd3f95	**LTDDipoContract.sol**
00db255283adf314d53801f80ecd10773f4dd6109bc2016e726292a81dd40e50	**LTDMasterChef.sol**
470807e3b005492dd57aca37588b09a0b9ac089949776a8082809f568238fd01	**LTDMembershipNft.sol**
2731eedb832ff0aef6a54a79f502079e19b400d2d0ba707ee47f9e74a181c66d	**LTDMigiration.sol**
70648add455198735e1773f825e12a1282d5392b461803a3338ee1d4b01bee31	**LTDStaking.sol**
91ace210c17fb856ec2456ef2e577ecfc7acd7ff135d583088f8ac7dc2901167	**LTDStakingWhitelist.sol**
1bccaca568d465461e2e43b37485147e030c2c7f4408846d4dd4673bee267a1a	**LTDSwapStock.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 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 LiveTrade Smart Contracts was written in Solidity language, with the required version to be ^0.8.2.

2.1.1. LiveTradeMasterChef contract

This is the master contract of the LiveTrade Smart Contracts which extends Ownable contract. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time.

Contract Owner can add many pools to LiveTradeMasterChef contract. Then, users can deposit LP token into the pool to get LiveTrade reward token. Contract Owner can stop/resume any pools, users can not deposit to stopped pools but can still withdraw their LP token.

2.1.2. BaseNFTBountyStake contract

This is the base of membership NFT contract in the LiveTrade Smart Contracts which extends ERC721, ERC721Enumerable, Pausable, Ownable, ERC721Burnable and Multicall. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time. Token Owner can pause/unpause contract using Pausable contract, users can only transfer NFTs when the contract is not paused.

Users can acquire membership NFTs by exchanging LTD tokens during the NFT offering. The number of LTD tokens need to exchange for NFTs is different with each membership level and can be set by Token Owner. Users can redeem their membership NFTs for LTD tokens but have to bear a 1.25% fee.

2.1.3. LTDMigration contract

This is the token migration contract in the LiveTrade Smart Contracts which extends Ownable and ReentrancyGuard. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time.

Whitelisted users can deposit old LiveTrade tokens, do swap with a customizable ratio set by Contract Owner, then withdraw new LiveTrade tokens. Contract Owner can pause/unpause contract, users can only swap tokens when the contract is not paused. He can also withdraw all tokens (both old and new) from this contract at anytime.

2.1.4. LiveTradeStaking contract

This is the staking contract in the LiveTrade Smart Contracts which extends Ownable and ReentrancyGuard. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time.

Users can deposit staked token into the pool to get reward token after each block. Contract Owner can stop rewarding at anytime. He can also withdraw reward tokens in case of emergency. Any wrong ERC20 tokens sent to the contract by mistake can be recovered by Contract Owner except staked token (token which is used for staking).

2.1.5. LTDStakingNFTWhitelist contract

This is the NFT staking contract in the LiveTrade Smart Contracts which extends Ownable and ReentrancyGuard. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time.

Users can deposit whitelisted NFT into this contract. They can only claim back their NFT after locking time which is set by Contract Owner.

2.1.6. LTDSwapStock contract

This is the swap stock contract in the LiveTrade Smart Contracts which extends Pausable and ReentrancyGuard. The contract is used for swapping between stock token and another token which are set when contract deployed. The contract also set caller, maintainer, fundKeeper and feeCollector role.

First, caller calculate a quote from a price user selected and lock it in with a transaction in the contract. User then confirm the quote and transfer input tokens for payment to the contract. Finally, caller executes the transaction by taking fee and transferring output tokens to users. User can choose cancel the transaction and get the refund of input tokens only after confirmation and before the transaction is executed.

maintainer can pause/unpause the contract. When the contract is paused, all functions of the contract will be stopped except emergencyRecovery which fundKeeper can use to recover token and stock token from the contract.

2.1.7. BaseDipoLTD contract

This is the Digital Initial Private Offering contract in the LiveTrade Smart Contracts which extends Ownable and ReentrancyGuard. With Ownable, by default, Contract Owner is contract deployer, but he can transfer ownership to another address at any time. Contract Owner can withdraw HTD token at anytime in case of emergency.

Digital Initial Private Offering is a new solution dedicated by LiveTrade for SMBs to widen the range of investors to obtain fast and easy capital by leveraging blockchain technology and financial knowledge into one single system.

2.2. Findings

During the audit process, the audit team found some vulnerability issues in the given version of LiveTrade Smart Contracts.

LiveTrade fixed the code, according to Verichains's private report, in commit 9e87bbf83fd33d3f0962e8f3556ebebc71ad7a26.

2.2.1. LTDDipoContract.sol - claimTGE without checking for startTimeTGE CRITICAL

In claimTGE function, the contract does not check for startTimeTGE so users can claim token immediately after IDO. They can then use the tokens to add LP before TGE.

```
function setTimeTGE(uint256 _time, uint256 _tgeRelease) external onlyOwne…
  r {
    startTimeTGE = _time;

    TGE_RELEASE = _tgeRelease;
}

function claimTGE() external nonReentrant {
    require(userIsPaidIDO[msg.sender] == true, "You need paid IDO first");

    require(isPauseContract != true, "This contract is pause!");

    require(locks[msg.sender] > 0, "You're not buy this round");

    require(userIsClaimTGE[msg.sender] == false, "TGE is claimed");

    // calc token can claim TGE
    uint256 HTDAmount = locks[msg.sender].mul(TGE_RELEASE).div(100);

    // decrease remain
    userRemaining[msg.sender] -= HTDAmount;

    // transfer token to user
    htdToken.transfer(msg.sender, HTDAmount);

    // Set userIsClaimTGE
    userIsClaimTGE[msg.sender] = true;
    emit ClaimTGE(msg.sender, HTDAmount, block.timestamp);
}
```

RECOMMENDATION

Checking for time passed startTimeTGE when claimTGE.

UPDATES

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

2.2.2. Comparing boolean with true/false INFORMATIVE

Many places in the contract comparing booleans with true/false, which make their meaning harder to understand.

For example:

```
require(isPauseContract != true, "This contract is pause!");
```

Can be simplified into:

```
require(!isPauseContract, "This contract is pause!");
```

which is easier to read.

UPDATES

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

2.2.3. LTDDipoContract.sol - governanceRecoverUnsupported can withdraw any ERC20 Token INFORMATIVE

The function governanceRecoverUnsupported is used for recover unsupported tokens but Contract Owner can withdraw any token from contract at anytime.

```
/* ========== EMERGENCY ========== */
function governanceRecoverUnsupported(
    address _token,
    address _to,
    uint256 _amount
) external onlyOwner {
    // require(_token != address(HTD), "Token invalid");
    ERC20(_token).transfer(_to, _amount);
}
```

RECOMMENDATION

Consider allow recover any tokens except HTD after release time for a reasonable period.

UPDATES

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

2.2.4. LTDDipoContract.sol - Unnecessary nonReentrant in setOwnerIDO INFORMATIVE

The function setOwnerIDO only sets a variable, so it is unnecessary to use nonReentrant modifier.

```
function setOwnerIDO(address _owner) external nonReentrant onlyOwner {
    ownerDipo = _owner;
}
```

RECOMMENDATION

Consider removing nonReentrant modifier.

UPDATES

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

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Jan 31, 2022*	Private Report	Verichains Lab
1.1	*Feb 08, 2022*	Public Report	Verichains Lab

Table 2. Report versions history