SECURITY AUDIT OF

MUMMY FINANCE SMART CONTRACT

Public Report

Dec 14, 2023

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 Dec 14, 2023. We would like to thank the Mummy Finance 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 Mummy Finance 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 issue in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About Mummy Finance Smart Contract

The GO-TO solution for swapping and perpetual trading $BTC, $ETH, $FTM, $OP, $ARB, and much more.

1.2. Audit scope

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

It was conducted on commit `43c28c3e7270ec97a0d14445bd58104c06cf53ed` from git repository *https://github.com/mummy-finance/navigator-contracts*.

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

SHA256 Sum	File
75f2f4e24b9dc8ee233f5c6e3e33b43860d747bb5e97da8a3df08f23d7478582	OrderExecutor.sol

1.3. Audit methodology

Our security audit process includes four steps:

- Mechanism Design is reviewed to look for any potential problems.
- Source codes are scanned/tested for commonly known and more specific vulnerabilities using public and our in-house security analysis tool.
- Manual audit of the codes for security issues. The source code is manually analyzed to look for any potential problems.
- Set up a testing environment to debug/analyze found issues and verifies our attack PoCs.

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 functioning; creates a critical risk to the application; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the application with high impact; needs to be fixed with high priority.

SEVERITY LEVEL	DESCRIPTION
MEDIUM	A vulnerability that could affect the desired outcome of executing the application 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

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

2. AUDIT RESULT

2.1. Overview

The Mummy Finance Smart Contract was written in `Solidity` language, with the required version to be `^0.8.9`. The source code was written based on OpenZeppelin's library.

2.1.1. OrderExecutor.sol

The `OrderExecutor` contract extends the `Initializable` contract. Upon initialization, the contract will pass the addresses of other contracts, such as `PriceManager`, `PositionVault`, `OrderVault`, `Operators` and `LiquidateVault`, for their initialization.

This contract allows operators to execute orders and positions that have attained entry, take profit, stop loss, open and liquidate Positions, and other orders. These execution functions will use the `priceManager.setPrice()` function to update the current and past prices of assets, and the caller of these functions must be an operator with at least level 1 on the Contract `Operators` side.

2.2. Findings

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

APPENDIX

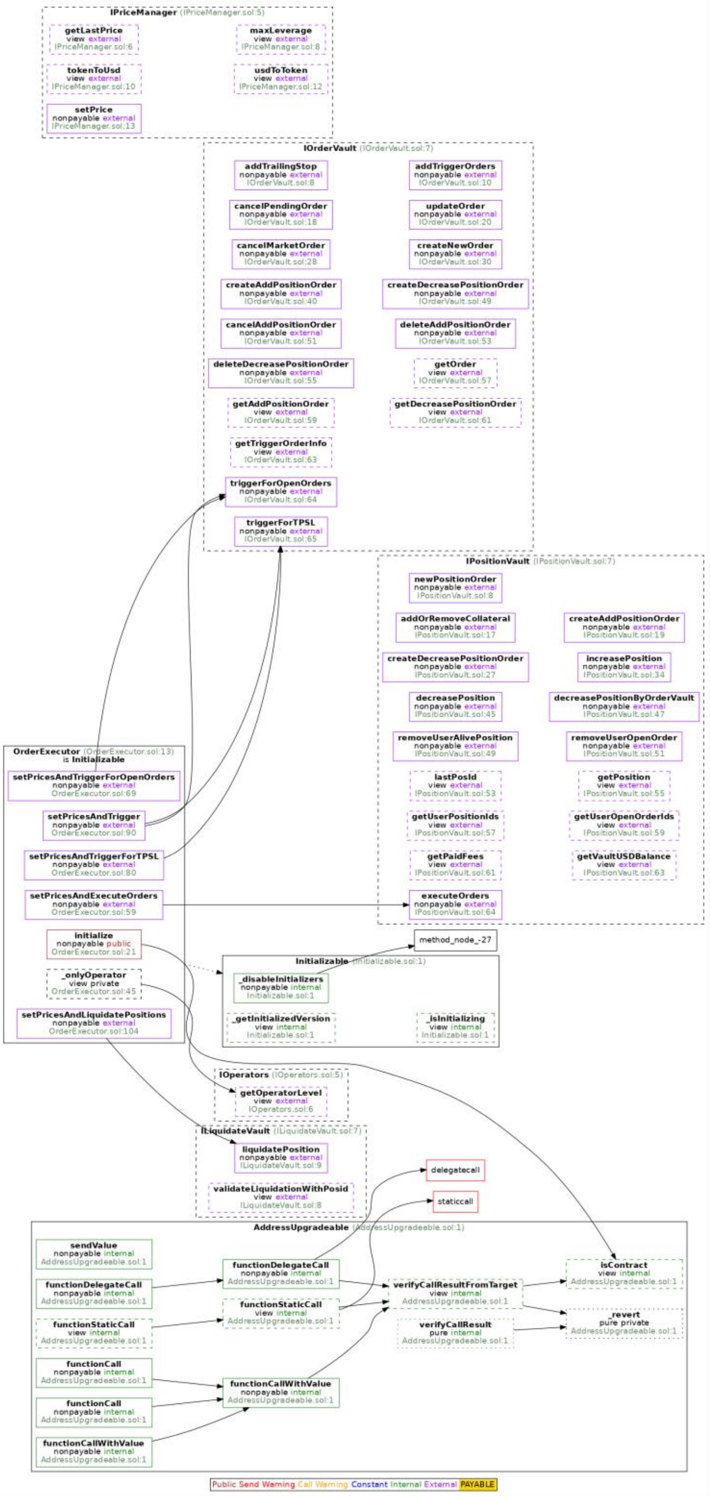
IPriceManager (IPriceManager.sol:5)
getLastPrice view external IPriceManager.sol:6
maxLeverage view external IPriceManager.sol:8
tokenToUsd view external IPriceManager.sol:10
usdToToken view external IPriceManager.sol:12
setPrice nonpayable external IPriceManager.sol:13
IOrderVault (IOrderVault.sol:7)
addTrailingStop nonpayable external IOrderVault.sol:8
addTriggerOrders nonpayable external IOrderVault.sol:10
cancelPendingOrder nonpayable external IOrderVault.sol:18
updateOrder nonpayable external IOrderVault.sol:20
cancelMarketOrder nonpayable external IOrderVault.sol:28
createNewOrder nonpayable external IOrderVault.sol:30
createAddPositionOrder nonpayable external IOrderVault.sol:40
createDecreasePositionOrder nonpayable external IOrderVault.sol:49
cancelAddPositionOrder nonpayable external IOrderVault.sol:51
deleteAddPositionOrder nonpayable external IOrderVault.sol:53
deleteDecreasePositionOrder nonpayable external IOrderVault.sol:55
getOrder view external IOrderVault.sol:57
getAddPositionOrder view external IOrderVault.sol:59
getDecreasePositionOrder view external IOrderVault.sol:61
getTriggerOrderInfo view external IOrderVault.sol:63
triggerForOpenOrders nonpayable external IOrderVault.sol:64
triggerForTPSL nonpayable external IOrderVault.sol:65
IPositionVault (IPositionVault.sol:7)
newPositionOrder nonpayable external IPositionVault.sol:8
addOrRemoveCollateral nonpayable external IPositionVault.sol:17
createAddPositionOrder nonpayable external IPositionVault.sol:19
createDecreasePositionOrder nonpayable external IPositionVault.sol:27
increasePosition nonpayable external IPositionVault.sol:34
decreasePosition nonpayable external IPositionVault.sol:45
decreasePositionByOrderVault nonpayable external IPositionVault.sol:47
removeUserAlivePosition nonpayable external IPositionVault.sol:49
removeUserOpenOrder nonpayable external IPositionVault.sol:51
lastPosId view external IPositionVault.sol:53
getPosition view external IPositionVault.sol:55
getUserPositionIds view external IPositionVault.sol:57
getUserOpenOrderIds view external IPositionVault.sol:59
getPaidFees view external IPositionVault.sol:61
getVaultUSDBalance view external IPositionVault.sol:63
executeOrders nonpayable external IPositionVault.sol:64
OrderExecutor (OrderExecutor.sol:13) is Initializable
setPricesAndTriggerForOpenOrders nonpayable external OrderExecutor.sol:69
setPricesAndTrigger nonpayable external OrderExecutor.sol:90
setPricesAndTriggerForTPSL nonpayable external OrderExecutor.sol:80
setPricesAndExecuteOrders nonpayable external OrderExecutor.sol:59
initialize nonpayable public OrderExecutor.sol:21
_onlyOperator view private OrderExecutor.sol:45
setPricesAndLiquidatePositions nonpayable external OrderExecutor.sol:104
method_node_-27
Initializable (Initializable.sol:1)
_disableInitializers nonpayable internal Initializable.sol:1
_getInitializedVersion view internal Initializable.sol:1
_isInitializing view internal Initializable.sol:1
IOperators (IOperators.sol:5)
getOperatorLevel view external IOperators.sol:6
ILiquidateVault (ILiquidateVault.sol:7)
liquidatePosition nonpayable external ILiquidateVault.sol:9
validateLiquidationWithPosid view external ILiquidateVault.sol:8
delegatecall
staticcall
AddressUpgradeable (AddressUpgradeable.sol:1)
sendValue nonpayable internal AddressUpgradeable.sol:1
functionDelegateCall nonpayable internal AddressUpgradeable.sol:1
functionStaticCall view internal AddressUpgradeable.sol:1
functionCall nonpayable internal AddressUpgradeable.sol:1
functionCall nonpayable internal AddressUpgradeable.sol:1
functionCallWithValue nonpayable internal AddressUpgradeable.sol:1
functionDelegateCall nonpayable internal AddressUpgradeable.sol:1
functionStaticCall view internal AddressUpgradeable.sol:1
functionCallWithValue nonpayable internal AddressUpgradeable.sol:1
verifyCallResultFromTarget view internal AddressUpgradeable.sol:1
verifyCallResult pure internal AddressUpgradeable.sol:1
isContract view internal AddressUpgradeable.sol:1
_revert pure private AddressUpgradeable.sol:1
Public Send Warning Call Warning Constant Internal External PAYABLE

Image 1. Mummy Finance Smart Contract call graph

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Dec 14,2023*	Public Report	Verichains Lab

Table 2. Report versions history