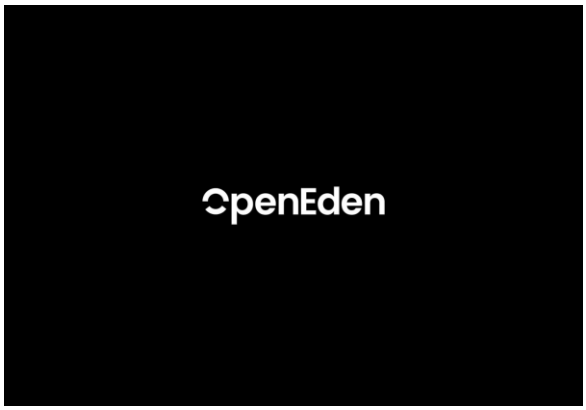
verichains

*SECURITY AUDIT OF*

# OPENEDEN SMART CONTRACT




**Public Report**

*Mar 29, 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 Mar 29, 2023. We would like to thank the OpenEden 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 OpenEden 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 OpenEden Smart Contract

OpenEden has launched the first smart contract vault managed by a regulated financial institution that offers 24/7 and direct access to U.S. Treasury Bills (T-Bills).

Stablecoin holders can mint TBILL tokens via the OpenEden TBILL Vault to earn the U.S. risk-free rate.

## 1.2. Audit scope

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

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

| SHA256 Sum | File |
|---|---|
| 51b1723cfe13e835393067ec250748c65c9db8e29064df262aa92db59ea3f490 | BaseVault.sol |
| 540b0d51f2e8fc0428ed38feec6a0e93eea9f573be425e58fd31af0617f4fb6a | OpenEdenVault.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

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

The OpenEden Smart Contract was written in `Solidity` language, with the required version to be `^0.8.4`. The source code was written based on OpenZeppelin's library.

### 2.1.1. BaseVault.sol

BaseVault.sol extends `Ownable` contract. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time. Only the owner can change the on-chain data of this contract. The owner can change parameters such as `transactionFee`, `firstDeposit`, `minDeposit`, `maxDeposit`, `maxWithdraw`, `targetReservesLevel`, `onchainServiceFeeRate`, and `offchainServiceFeeRate`.

### 2.1.2. OpenEdenVault.sol

OpenEdenVault.sol extends `ERC4626Upgradeable`, `ReentrancyGuardUpgradeable`, `PausableUpgradeable`, `ChainlinkAccessor`, and `AccessControl` contracts.

This contract is an ERC4626 token contract which is a standard to optimize and unify the technical parameters of yield-bearing vaults. It provides a standard API for tokenized yield-bearing vaults that represent shares of a single underlying ERC-20 token. ERC-4626 also outlines an optional extension for tokenized vaults utilizing ERC-20, offering basic functionality for depositing, withdrawing tokens and reading balances.

The users with `DEFAULT_ADMIN_ROLE` or `OPERATOR_ROLE` can pause/unpause the contract using the `PausableUpgradeable` contract. Users can only deposit or withdraw tokens when the contract is not paused. The Admin or the Operator users can transfer any underlying assets to the treasury address by the `fundTBillPurchase()` function and can claim onchain/offchain service fees.

When the user calls the `deposit()` and `withdraw()` functions, this contract will send a request to Chainlink, and Chainlink will call the `fulfill()` callback function to process the user's deposit/withdrawal order.

The `fulfill()` function performs the corresponding actions such as `DEPOSIT`, `WITHDRAW`, `WITHDRAW_QUEUE`, `EPOCH_UPDATE`. When the amount of assets in the contract is not enough to execute the withdraw order, the remaining amount will be put into `QueueWithdrawal`, until the Admin or Operator calls the `processWithdrawalQueue()` action to continue withdrawing the remaining funds for users. With the `EPOCH_UPDATE` action, the contract will update values such as `onchainFeeRate`, `offchainFeeRate`, `_onchainFee`, `_offchainFee` and increase the `_epoch` by 1 unit. In addition, the `fulfill()` function will update the value of the `_latestOffchainAsset` variable,

which will affect the calculation of the amount of assets available at the time of deposit and withdrawal, and it will also modify the value of the `_offchainFee` variable.

There are certain conditions for depositing, withdrawing, and transferring tokens, such as the user has completed KYC and is not banned. In addition, there are other conditions for transferring tokens, and all of this logic is included in the KycManager contract.

## 2.2. Findings

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

## 2.3. Additional notes and recommendations

### 2.3.1. OpenEdenVault.sol - The `previewDeposit()` and `previewWithdraw()` functions may differ from the actual amount of deposit or withdrawal that the user receives INFORMATIVE

If OpenEden Smart Contract intends to support EOA account access directly, **you should consider adding an additional function call for deposit/withdraw with the means to accommodate slippage loss or unexpected deposit/withdrawal limits, since they have no other means to revert the transaction if the exact output amount is not achieved.**

The methods totalAssets, convertToShares and convertToAssets are estimates useful for display purposes, and do not have to confer the exact amount of underlying assets their context suggests.

The preview methods return values that are as close as possible to exact as possible. For that reason, they are manipulable by altering the on-chain conditions and are not always safe to be used as price oracles. This specification includes convert methods that are allowed to be inexact and therefore can be implemented as robust price oracles. For example, it would be correct to implement the convert methods as using a time-weighted average price in converting between assets and shares.

```
function _convertToAssets(
    uint256 shares,
    MathUpgradeable.Rounding rounding
) internal view override returns (uint256 assets) {
    uint256 supply = totalSupply();
    return
        (supply == 0)
            ? _initialConvertToAssets(shares, rounding)
            : shares.mulDiv(assetsAvailable(), supply, rounding);
}
```

### 2.3.2. BaseVault.sol - Redundancy of the event function INFORMATIVE

The event `SetFirsetDeposit()` is not used.

**RECOMMENDATION**

Remove this event.

# 3. VERSION HISTORY

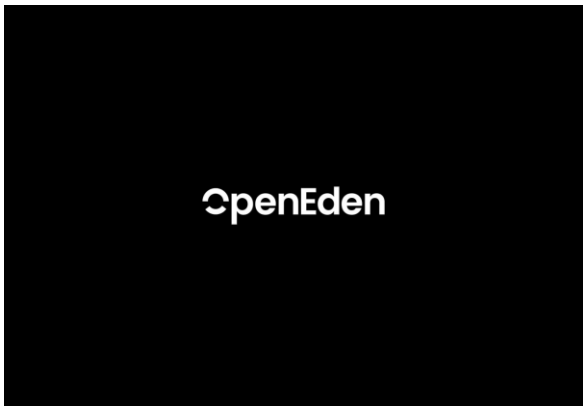
| Version | Date | Status/Change | Created by |
|---|---|---|---|
| **1.0** | *Mar 29,2023* | Public Report | Verichains Lab |

*Table 2. Report versions history*

verichains

*SECURITY AUDIT OF*

# OPENEDEN SMART CONTRACT




**Public Report**

*Mar 31, 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 Mar 31, 2023. We would like to thank the OpenEden 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 OpenEden 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 OpenEden Smart Contract

OpenEden has launched the first smart contract vault managed by a regulated financial institution that offers 24/7 and direct access to U.S. Treasury Bills (T-Bills).

Stablecoin holders can mint TBILL tokens via the OpenEden TBILL Vault to earn the U.S. risk-free rate.

## 1.2. Audit scope

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

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

| SHA256 Sum | File |
|---|---|
| 51b1723cfe13e835393067ec250748c65c9db8e29064df262aa92db59ea3f490 | BaseVault.sol |
| 540b0d51f2e8fc0428ed38feec6a0e93eea9f573be425e58fd31af0617f4fb6a | OpenEdenVault.sol |
| 15d78edd9cada841fd20501bac68877dfc72d7ff33c84d84de2cdfad4da27dec | KycManager.sol |
| 5d770938e7dce60384755fbfcc15a54a7d2fc49346fd29766e23448438743a9a | ChainlinkAccessor.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

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

The OpenEden Smart Contract was written in `Solidity` language, with the required version to be `^0.8.4`. The source code was written based on OpenZeppelin's library.

### 2.1.1. BaseVault.sol

BaseVault.sol extends `Ownable` contract. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time. Only the owner can change the on-chain data of this contract. The owner can change parameters such as `transactionFee`, `firstDeposit`, `minDeposit`, `maxDeposit`, `maxWithdraw`, `targetReservesLevel`, `onchainServiceFeeRate`, and `offchainServiceFeeRate`.

### 2.1.2. KycManager.sol

KycManager.sol extends `Ownable` contract. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time. Only the owner can change the on-chain data of this contract.

There are 3 types of KYC: `NON_KYC`, `US_KYC`, `GENERAL_KYC`. The Owner can assign any address to one of these 3 KYC types and can toggle Strict mode. Additionally, the Owner can also decide whether a specific address can be included in the banned/unbanned list.

### 2.1.3. ChainlinkAccessor.sol

When the contract deployer of the `OpenEdenVault` contract calls the `initialize()` function, it will call the `init()` function inside `ChainlinkAccessor` contract, which will initialize the initial values for `ChainlinkParameters`, including `fee`, `jobId`, `urlData`, `pathToOffchainAssets`, `pathToTotalOffchainAssetAtLastClose`, and also initialize the `chainlinkOracle` and `chainlinkToken` addresses. The Admin can change the values in `ChainlinkParameters`.

### 2.1.4. OpenEdenVault.sol

OpenEdenVault.sol extends `ERC4626Upgradeable`, `ReentrancyGuardUpgradeable`, `PausableUpgradeable`, `ChainlinkAccessor`, and `AccessControl` contracts.

This contract is an ERC4626 token contract which is a standard to optimize and unify the technical parameters of yield-bearing vaults. It provides a standard API for tokenized yield-bearing vaults that represent shares of a single underlying ERC-20 token. ERC-4626 also outlines an optional extension for tokenized vaults utilizing ERC-20, offering basic functionality for depositing, withdrawing tokens and reading balances.

The users with `DEFAULT_ADMIN_ROLE` or `OPERATOR_ROLE` can pause/unpause the contract using the `PausableUpgradeable` contract. Users can only deposit or withdraw tokens when the contract is not paused. The Admin or the Operator users can transfer any underlying assets to the treasury address by the `fundTBillPurchase()` function and can claim onchain/offchain service fees.

When the user calls the `deposit()` and `withdraw()` functions, this contract will send a request to Chainlink, and Chainlink will call the `fulfill()` callback function to process the user's deposit/withdrawal order.

The `fulfill()` function performs the corresponding actions such as `DEPOSIT`, `WITHDRAW`, `WITHDRAW_QUEUE`, `EPOCH_UPDATE`. When the amount of assets in the contract is not enough to execute the withdraw order, the remaining amount will be put into `QueueWithdrawal`, until the Admin or Operator calls the `processWithdrawalQueue()` action to continue withdrawing the remaining funds for users. With the `EPOCH_UPDATE` action, the contract will update values such as `onchainFeeRate`, `offchainFeeRate`, `_onchainFee`, `_offchainFee` and increase the `_epoch` by 1 unit. In addition, the `fulfill()` function will update the value of the `_latestOffchainAsset` variable, which will affect the calculation of the amount of assets available at the time of deposit and withdrawal, and it will also modify the value of the `_offchainFee` variable.

There are certain conditions for depositing, withdrawing, and transferring tokens: Users who want to deposit and withdraw must be users who have been KYC'd within the `KycManager` contract (either `US_KYC` or `GENERAL_KYC`), and not be on the banned list. Additionally, there are 3 other conditions when users want to transfer tokens:

- If the sender and/or recipient are on the banned list, they will not be able to transfer/receive tokens.
- If the Strict mode of the KycManager contract is enabled, only the sender/receiver who has been KYCed can send/receive.
- If the KYC type of the sender is `US_KYC`, then the recipient must also be KYCed.

## 2.2. Findings

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

## 2.3. Additional notes and recommendations

### 2.3.1. ChainlinkAccessor.sol - Contract redundancy INFORMATIVE

The contract `hardhat/console.sol` is imported but not used.

**RECOMMENDATION**

Remove the `hardhat/console.sol` contract.

### 2.3.2. OpenEdenVault.sol - The `previewDeposit()` and `previewWithdraw()` functions may differ from the actual amount of deposit or withdrawal that the user receives INFORMATIVE

If OpenEden Smart Contract intends to support EOA account access directly, **you should consider adding an additional function call for deposit/withdraw with the means to accommodate slippage loss or unexpected deposit/withdrawal limits, since they have no other means to revert the transaction if the exact output amount is not achieved.**

The methods totalAssets, convertToShares and convertToAssets are estimates useful for display purposes, and do not have to confer the exact amount of underlying assets their context suggests.

The preview methods return values that are as close as possible to exact as possible. For that reason, they are manipulable by altering the on-chain conditions and are not always safe to be used as price oracles. This specification includes convert methods that are allowed to be inexact and therefore can be implemented as robust price oracles. For example, it would be correct to implement the convert methods as using a time-weighted average price in converting between assets and shares.

```
function _convertToAssets(
    uint256 shares,
    MathUpgradeable.Rounding rounding
) internal view override returns (uint256 assets) {
    uint256 supply = totalSupply();
    return
        (supply == 0)
            ? _initialConvertToAssets(shares, rounding)
            : shares.mulDiv(assetsAvailable(), supply, rounding);
}
```

### 2.3.3. BaseVault.sol, ChainlinkAccessor.sol - The redundant comment and event function INFORMATIVE

- There is an unnecessary `// TODO` comment at line 94 in the ChainlinkAccessor contract.
- The event `SetFirsetDeposit()` is not used in BaseVault contract.

**RECOMMENDATION**

Remove both.

# 3. VERSION HISTORY

| Version | Date | Status/Change | Created by |
|---|---|---|---|
| **1.0** | *Mar 29,2023* | Public Report | Verichains Lab |
| **1.1** | *Mar 31,2023* | Public Report | Verichains Lab |

*Table 2. Report versions history*