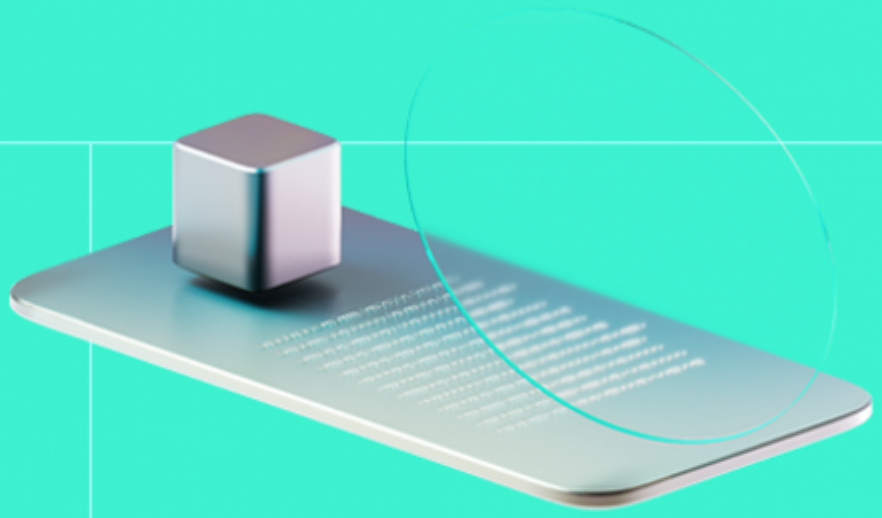
HACKEN

Smart Contract Code Review And Security Analysis Report

Customer: OpenEden

Date: 09/02/2024



We express our gratitude to the OpenEden team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Openeden, is an project that manages the TBILL stable ERC20 tokens. Users can deposit USDC to mint TBILL tokens, entitling them to redeem assets in proportion to their TBILL holdings

Platform: EVM

Language: Solidity

Tags: ERC20 stable coin.

Timeline: 31/01/2024 - 09/02/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/OpenEdenHQ/openeden.vault.v2.audit/
Commit	d09f86cb9827242dc6e76033a60c7a464aebe27d

Audit Summary

10/10	10/10	89.42%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 9.6/10

The system users should acknowledge all the risks summed up in the risks section of the report

2	1	1	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	1
Low	1

Vulnerability	Status
F-2024-0760 - The Contract Has The Function Which Allows To Withdraw Tokens Including The `underlying`	Accepted
F-2024-0719 - Authorization Using tx.origin	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for OpenEden
Audited By	Maksym Fedorenko
Approved By	Grzegorz Trawinski
Website	https://openeden.com/
Changelog	05/02/2024 - Preliminary Report; 09/02/2024 - Second Report

Table of Contents

System Overview **6**

Privileged Roles 6

Executive Summary **8**

Documentation Quality 8

Code Quality 8

Test Coverage 8

Security Score 8

Summary 8

Risks **9**

Findings **10**

Vulnerability Details 10

Observation Details 14

Disclaimers 19

Appendix 1. Severity Definitions **20**

Appendix 2. Scope **21**

System Overview

Openeden, is an project that manages the TBILL stable ERC20 tokens. Users can deposit USDC to mint TBILL tokens, entitling them to redeem assets in proportion to their TBILL holdings. TBILL tokens are stored in whitelisted wallets, and the project's code governs deposit, withdrawal, and management functions, ensuring proper asset handling. It is built with the following contracts:

- Controller — The Controller contract provides mechanisms for pausing and unpausing specific operations (deposit and withdraw) in a system.
- FeeManager - The FeeManager contract provides mechanisms to manage various fee-related parameters in a system. This includes settings for transaction fees, deposit and withdrawal limits, management fee rates, and special considerations for weekends.
- OpenEdenVaultV3Impl - is an upgradeable vault contract designed for managing deposits and withdrawals, charging fees, integrating with KYC systems, and operating under specific time-based rules.
- TBillPriceOracle - Oracle contract provides a way to manage and update TBill prices with constraints on how much the price can deviate from previous values.
- Timelock - Imports TimelockController from Openzeppelin.
- OEPausable - The contract is designed to introduce “pausing” functionality into a contract by inheritance. This pausing mechanism can be utilized for emergency scenarios or other use cases to temporarily halt certain operations of a contract.

Privileged roles

- The DEFAULT_ADMIN_ROLE of the Controller contract can:
  - Pause and unpause deposits and withdrawals
- The OPERATOR_ROLE of the Controller contract can:
  - Pause and unpause deposits and withdrawals
- The Owner of the FeeManager contract can:
  - The owner can set various fee-related parameters like transaction fees, deposit and withdrawal limits, etc.
  - The owner inherits the capabilities provided by the OpenZeppelin's Ownable contract, such as the ability to transfer ownership or renounce ownership.
- The Owner of the OpenEdenVaultV3Impl can:
  - Set the treasury for the vault.
  - Set the treasury specific to 'q' (qTreasury).
  - Toggle whether the USDC/USD price is fixed.
  - Set various addresses, such as FeeManager, KycManager,
  - Operator, USDC Price Feed, TBill Price Feed, and Controller.
  - Authorize contract upgrades.
  - Upgrade the contract.
- The Operator of the OpenEdenVaultV3Impl can:
  - Initiate off-ramp operations to transfer underlying assets to designated treasuries.
  - Process the withdrawal queue.
  - Update the epoch and set whether it is a weekend.
  - Claim the service fee.
- The DEFAULT_ADMIN_ROLE of the TBillPriceOracle can:
  - Grant and revoke the OPERATOR_ROLE.
  - Can update the maximum price deviation.
  - Can manually update the close NAV price.
- The OPERATOR_ROLE of the TBillPriceOracle can:
  - Can update the price.
  - Can update the close NAV price.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.
- NatSpec is provided.

Code quality

The total Code Quality score is **10** out of **10**.

Test coverage

Code coverage of the project is **89.42%** (branch coverage).

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **1** medium, and **1** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.6**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The logic of the OpenEdenVaultV3 contract might be upgraded any time by the Admin.
- The Epoch might be updated unlimited amount of times by the operator.
- At any given time, the owner or operator holds the capability to pause both withdrawals and deposits.
- The transaction fees within the system can be configured to any value, including 100%, which implies that users may receive nothing in return when attempting to deposit or redeem tokens. The fee is calculated and collected when the operator executes the withdrawal queue, based on the latest fee rate, not the rate at the time of user redemption.
- The TBILL price used in calculations is provided by an oracle through an off-chain mechanism and the implementation to sustain stable coin mechanism is also handled off-chain.
- The owner has the authority to withdraw any token including USDC from the contract.
- Only the operator has the authority to execute redemption requests, and the timing of execution is determined by the operator's discretion.
- The Treasury, KycManager and TBillPriceOracle contracts are beyond the scope of this audit (please navigate to “Appendix 2. Scope” for detailed information regarding the scope). The reliability of these contracts cannot be confirmed.
- The system employs a KYC process. There is a potential risk where a user, after receiving KYC approval and depositing tokens, could be subsequently banned. This would result in their tokens being permanently locked in their account.
- The redemption process relies on a centralized operator's discretion, and any issues or risks related to this operator's actions, such as insufficient USDC balance or centralization concerns, can result in users being unable to access their deposited TBILL tokens or the promised USDC tokens, potentially compromising the trustworthiness of the redemption process.
- There exists a risk where the backend system could fail to accurately set the isWeekend flag, which is crucial for enforcing the intended fee structure and deposit limits. In this case, users might be able to deposit funds under incorrect fee structures.
- The Open Eden operators can ban users, which may result in declining withdrawal requests if a user has already deposited funds. This mechanism was introduced to protect against withdrawal queue block due to USDC blacklisting mechanism.

Findings

Vulnerability Details

F-2024-0719 - Authorization Using tx.origin - Medium

Description:

Using `tx.origin` authorization in the `onlyOperator` modifier in the smart contract `OpenEdenVaultV3Impl.sol`, introduces a security vulnerability by not accurately identifying the actual initiator of a transaction. Reliance on tx.origin severely limits the functionality with Multisig or account abstraction wallets, as it cannot differentiate between the original sender and the contract calling another contract.

```
// only operator can call this function
modifier onlyOperator(address _sender) {
require(tx.origin == _sender, "permission denied");
_;
}
```

The issue might cause the phishing attack if the operator calls the malicious contract which might later invoke the functions on the `OpenEdenVaultV3Impl` contract to commit malicious actions within the next functions: `redeemL()`, `cancel()`, `offRamp()`, `offRampQ()`, `processWithdrawalQueue()`, `updateEpoch()`, `setWeekendFlag()`, `claimServiceFee()`, The issue might also affect the integrity with the Leverage protocol (which is out of scope).

This issue seriously compromises security and restricts the adaptability of the application, making the smart contract incompatible with enhanced Ethereum wallet structures and leading to potential breaches.

Assets:

- OpenEdenVaultV3Impl.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Fixed

Classification

Severity: Medium

Impact:

Likelihood [1-5]: 3

Impact [1-5]: 3

Exploitability [1-2]: 1

Complexity [0-2]: 0

Final Score: 3 (Medium)

Recommendations

Recommendation:

To mitigate the security risks associated with using `tx.origin` for authorization, it is recomended to transition to the use of `msg.sender`. Unlike `tx.origin`, `msg.sender` refers to the immediate sender of the current call, which ensures that the

authorization check is based on the originating entity of the current interaction with the contract.

Remediation (revised commit: **d09f86c):** The code was updated to utilize the `msg.sender` global variable for operators validation.

Evidences

PoC (steps)

Reproduce:

1. The `address1` is specified as an operator.
2. The attacker constructs malicious contract that interacts with the `OpenEdenVaultV3Impl.sol` contract.
3. The user who controls the `address1` calls the malicious contract, e.g. as a result of phishing attack.
4. The malicious contract calls the function which leads to the crucial state change of the `OpenEdenVaultV3Impl.sol` contract.

F-2024-0760 - The Contract Has The Function Which Allows To Withdraw Tokens Including The `underlying` - Low

Description:

The `OpenEdenVaultV3Impl.sol` contract has the function `offRampQ` which is created to transfer any unexpected token from the contract to the treasury contract with the `qTreasury` address, but the function also allows to transfer the `underlying` tokens which are supposed to be USDC. The `offRampQ` and `offRamp` function does not check if the contract has sufficient tokens to cover the `unClaimedFee` after the withdrawal.

This violates the requirement specified in the NatSpec and might lead to the DoS of the `claimServiceFee()` function if the `OpenEdenVaultV3Impl` contract has insufficient `underlying` balance for `unClaimedFee` withdrawal.

Assets:

- OpenEdenVaultV3Impl.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Accepted

Classification

Severity: Low

Impact:

Likelihood [1-5]: 3

Impact [1-5]: 3

Exploitability [1-2]: 2

Complexity [0-2]: 0

Final Score: 1,73 (Low)

Recommendations

Recommendation:

Add the conditional statement to disallow transferring the `underlying` tokens form the contract using the `offRampQ` function. Add validation to the `offRamp` function to disallow transfer if the contract does not have sufficient funds to cover unclaimed fees `unClaimedFee`.

Remediation: The Open Eden team accepted the issue, with the comment: the `offRampQ` function will transfer to qTreasury (quarantine treasury) and `offRamp` will transfer to treasury, the destinations are different and their events will be used by the backend server to take different actions.

Observation Details

F-2024-0704 - Unrestricted Fee Configuration - Info

Description:

The system owner possesses the authority to set transaction fees `setTxFeeHolidayWithdrawPct()`, `setTxFeeHolidayDepositPct()`, `setTxFeeWorkdayWithdrawPct()`, `setTxFeeWorkdayDepositPct()` for both weekends and weekdays to any value, including 100%. A fee rate of 100% implies that users will receive nothing when attempting to deposit or redeem tokens.

Moreover, charging fees for reedem operation does not happen in the redeem function immediately. The fee is taken when the operator process the withdrawal. If the fee is changed during this period of time, users will pay different amount of fees than promised.

This unrestricted control over fee configuration by the owner may lead to the imposition of excessive fees, resulting in users receiving no tokens for their transactions and potential dissatisfaction with the system. It introduces the risk of unfair and unjust fee charging and a potential loss of trust in the platform.

Assets:

- feeManager.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Accepted

Classification

Impact:

Likelihood [1-5]: 2

Impact [1-5]: 3

Exploitability [1-2]: 2

Complexity [0-2]: 0

Final Score: 1.6

Recommendations

Recommendation:

Set reasonable boundaries for the transaction fees and mention these max and min (if there is) limits in the documentation. Additionally, calculate and apply the fee within the redeem function based on the current fee rate, rather than collecting the fee after a user places a redemption order.

Remediation: The Open Eden team decided to not fix this observation due to commercial reasons.

F-2024-0720 - Missing Checks For `address(0)` - Info

Description:

In Solidity, the Ethereum address `0x0000000000000000000000000000000000000000` is known as the "zero address". This address has significance because it is the default value for uninitialized address variables and is often used to represent an invalid or non-existent address. The Missing zero address control issues arise when a Solidity smart contract `PartnerShip.sol` does not properly check or prevent interactions with the zero address in `createPartnerShip()` `_parent` parameter, `getParent()` `_child` parameter, leading to unintended behavior.

For instance, a contract might allow setting the zero address without any checks leading to unintended behaviour .

Assets:

- PartnerShip.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Accepted

Recommendations

Recommendation:

It is strongly recommended to implement checks to prevent the zero address from being set during the initialization of contracts. This can be achieved by adding require statements that ensure address parameters are not the zero address.

Remediation: The Open Eden acknowledged this observation.

F-2024-0745 - Redundant `_msgSender()`, Meta-Transactions Not Implemented - Info

Description:

The `_msgSender()` function is needed to handle the meta transactions, in the OpenZeppelin library, it is used to support the development of the libraries which might be used with the contracts with the specified TrustedForwarder or to be used in such contracts directly.

However, the current implementation is not a library and does not rewrite the `_msgSender()` function to support meta transactions. This leads to the redundancy, because the system do not utilize `_msgSender()` features and used only as a substitute for `msg.sender`.

Assets:

- OpenEdenVaultV3Impl.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Accepted

Recommendations

Recommendation:

If `_msgSender()` is not going to be used in current implementation to support the meta transactions it is recommended to use `msg.sender` to prevent unforeseen issues and misinterpretations that may occur after the future updates.

Remediation: The Open Eden acknowledged this observation.

F-2024-0759 - Missing Failover Mechanism to Unlock Withdrawal Queue Blocked by Non-Banned Users - Info

Description:

The `cancel()` function plays a critical role in managing withdrawal requests for banned users, specifically those who have been blacklisted by the internal mechanism of USDC. These requests must be canceled because attempting to transfer funds to a blacklisted account results in transaction reversal and queue processing blockage, preventing other users from withdrawing their funds.

While the `cancel()` function is effective in addressing some aspects of the queue blocking issue, it does not provide a comprehensive solution, as it is tailored exclusively to banned users. Several concerns persist regarding non-blocked users:

1. Denial of Service incidents may not always originate from banned users. Various scenarios, such as changes in Know Your Customer (KYC) status for the withdrawal receiver between the redemption and queue processing stages, can also lead to disruptions.
2. A separate issue pertains to the fees applied during the withdrawal processing. Users might still encounter discrepancies between the promised or agreed-upon fee amounts and the actual fees deducted. In exceptionally rare cases, if fees are altered between redemption and withdrawal queue processing, a user's withdrawal request could fail, causing queue blockages.

Assets:

- OpenEdenVaultV3Impl.sol [https://github.com/OpenEdenHQ/openeden.vault.v2.audit/commit/8cf7b346195ea00aa5012665a81b3889ad15da9d]

Status: Accepted

Recommendations

Recommendation: Implementing a failover mechanism is strongly advised to address situations where a valid (non-banned) user's withdrawal request experiences a reversal, resulting in the blocking of the withdrawal queue.

Remediation: The Open Eden acknowledged this observation.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details - initial

Repository	https://github.com/OpenEdenHQ/openeden.vault.v2.audit/
Commit	fdfe7a34f65a2a026632054c69e5b1a453c658a4
Whitepaper	None
Requirements	https://docs.openeden.com/treasury-bills-vault/introduction
Technical Requirements	https://docs.openeden.com/treasury-bills-vault/introduction

Scope Details - second

Repository	https://github.com/OpenEdenHQ/openeden.vault.v2.audit/
Commit	d09f86cb9827242dc6e76033a60c7a464aebe27d
Whitepaper	None
Requirements	https://docs.openeden.com/treasury-bills-vault/introduction
Technical Requirements	https://docs.openeden.com/treasury-bills-vault/introduction

Contracts in Scope

./contracts/feeManager.sol

./contracts/OpenEdenVaultV3Impl.sol

./contracts/interfaces/IPartnerShip.sol

./contracts/PartnerShip.sol