RISK AUDIT

for

PB&J CONSULTING

on

Jun 26, 2025

FIDESIUM

Executive Summary

Report

22
/100

TOTAL
Low risk
June 26, 2025

Abstract

Fidesium's automated risk assessment service was requested to perform a risk posture audit on TriviTourney **contracts**

Repository Link: https://github.com/PBJ-JWeb3/vesting-contracts

Initial Commit Hash:

`9244b56279a9d3a09ef6e17baacc2938260f4c51`

Issue Summary

Critical	High	Medium	Low	Info
0 Issues	4 Issues	7 Issues	1 Issues	0 Issues

Caveats

PBJ's codebase is generally well written, but does incur a handful of flaws.

Test Approach

Fidesium performed both Whitebox and Blackbox testing, as per the scope of the engagement, and relied on automated security testing.

Methodology

The assessment methodology covered a range of phases and employed various tools, including but not limited to the following:

- Mapping Content and Functionality of API
- Application Logic Flaws
- Access Handling
- Authentication/Authorization Flaws
- Brute Force Attempt
- Input Handling
- Source Code Review
- Fuzzing of all input parameter
- Dependency Analysis

Severity Definitions

Critical	The issue can cause large economic losses, large-scale data disorder or loss of control of authority management.
High	The issue puts users' sensitive information at risk or is likely to lead to catastrophic financial implications.
Medium	The issue puts a subset of users' sensitive information at risk, reputation damage or moderate financial impact.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is low-impact to the client's business.
Informational	The issue does not pose an immediate risk but is relevant to security best practices or defence in Depth.

Risk Issues

Vulnerability	Description	Risk	Probability	Status
Missing Oracle Validation	`OTCMarketplace` does not check if the oracle's data is valid.	High	Low	Active
Missing Oracle Validation: Stale Data	`OTCMarketplace` does not check if the oracle's data is stale and `_expiry` is in the future.	High	Low	Active
Centralization	The `owner` has significant modification rights over the contracts and their state.	Medium	Medium	Active
One step ownership transfer	Contracts rely on `Ownable` to manage ownership, which is not secure.	Medium	Medium	Active
Missing Access Control	`CoreVesting.batchReleaseTokens` does not check if the caller is the owner.	Medium	Medium	Active
Bespoke payment splitting calculation	`CoreVesting` implements bespoke payment splitting logic.	Medium	Low	Active
Logic Error: Hardcoded decimals	`OTCMarketplace.purchaseOTCDeal` hardcodes the decimals to `1e18`.	Medium	Low	Active
Logic Error: Hardcoded array indices on fee distribution	`CoreVesting.batchCreateVesting` hardcodes fees to first cliffTime	Medium	Low	Active
Reliance on Block Timestamp	`OTCMarketplace` relies on `block.timestamp`, which can be manipulated by miners.	Medium	Low	Acknowledged
Rounding error: Division in token amount calculations	`CoreVesting.batchCreateVesting` divides by token amounts.	Low	Medium	Active
Gas Ineffiency: Redundant reads and SStores in loop	`CoreVesting.releaseTokens` has inefficient reads and writes in a loop.	Info	Info	Active

Risk Overview

Team Risk

Low risk: 1

No issues found in founding team

Doxxing Status	Team Experience	Risk Summary
Public	Highly relevant	Low

Smart Contract Risks

Risk summary: 28

The contracts are well written, and secure with only a few minor issues..

Vulnerabilities Critical

Current scan criticals Clear

During this scan no critical security vulnerabilities were identified. The assessment covered all key components of the project, including smart contract logic, access controls, and potential attack vectors. While no critical issues were found, we recommend ongoing security monitoring and best practices to maintain the integrity and resilience of the system.

Vulnerabilities High

Missing Oracle Validation

Vulnerability severity: **High**

Vulnerability probability: **Low**

`OTCMarketplace` does not check if the oracle's data is valid.

A malicious oracle can manipulate the price of the token, and the contract will not be able to detect it, leading to unexpected results, economic attacks, or protocol failure.

Recommendations:

- Implement a validation check for the oracle's data.
- Implement a timelock for the oracle's data.
- Implement a fallback mechanism for the oracle's data.
- Implement a multi oracle system with aggregation
- Implement circuit breakers and price bounds
- Rely on TWAP oracles for price stability

Vulnerabilities High

Missing Oracle Validation: Stale Data

Vulnerability severity: **High**

Vulnerability probability: **Low**

`OTCMarketplace` does not check if the oracle's data is stale and `_expiry` is in the future.

Recommendations:

Implement a validation check for the oracle's data.

```
require(block.timestamp <= _expiry, "Oracle price data has expired");
require(_expiry <= block.timestamp + MIN_PRICE_AGE, "Oracle timestamp too far in future");
require(_expiry >= block.timestamp - MAX_PRICE_AGE, "Oracle price data too old");
```

Additionally implement nonce based validation to prevent replay attacks.

Vulnerabilities Medium

Centralization

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

The `owner` has significant modification rights over the contracts and their state.

Recommendations:

Ensure that these roles are tied to well maintained Multisig wallets, and consider implementing a timelock.

One Step Ownership Transfer

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

Contracts rely on `Ownable` to manage ownership, which is not secure.

The `Ownable` pattern is vulnerable to a one step ownership transfer. This exposes these contracts to accidental ownership transfer to malicious or invalid wallets.

Recommendations:

Implement `Ownable2Step` to drive a two step ownership transfer. This will require applying `Upgradeable` independently.

Missing Access Control

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

`CoreVesting.batchReleaseTokens` does not check if the caller is the owner.

Recommendations:

Ensure that the caller is the owner before releasing tokens.

Vulnerabilities Medium

Bespoke payment splitting calculation

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`CoreVesting` implements bespoke payment splitting logic.

This can introduce bugs, and is not recommended.

Recommendations:

Use a more standard payment splitting logic, such as OpenZeppelin's `PaymentSplitter`.

Vulnerabilities Medium

Logic Error: Hardcoded decimals

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

`OTCMarketplace.purchaseOTCDeal` hardcodes the decimals to `1e18`.

This will be incorrect for tokens with different decimals, such as USDC.

Recommendations:

Use the `decimals` function to get the decimals of the token.

```
uint8 tokenDecimals = IERC20(details.projectToken).decimals();
uint256 totalPrice = details.totalTokens * _currentPrice / (10 ** tokenDecimals);
```

Logic Error: Hardcoded array indices on fee distribution

Vulnerability severity: **Medium**

Vulnerability probability: **Low**

`CoreVesting.batchCreateVesting` hardcodes fees to first cliffTime

```
_createVesting(
    escrowWallet,
    params.projectToken,
    feeAmount,
    params.releaseInterval,
    params.numReleases,
    params.cliffTimes[0],
    params.cliffPercents[0],
    params.projectWallet,
    false,
    params.tgeAmount
);
```

Recommendations:

Aggregate wallet fees inside the loop based on params.cliffTimes[i]

Reliance on Block Timestamp

Vulnerability severity: **Medium**

Vulnerability probability: **Medium**

`OTCMarketplace` relies on `block.timestamp`, which can be manipulated by miners.

Recommendations:

Use a more secure timestamp source, such as a trusted oracle, or at least implement a compound time computation based on `block.timestamp` and `block.number` and `block.timestamp`

Vulnerabilities Low

Rounding error: Division in token amount calculations

Vulnerability severity: **Low**

Vulnerability probability: **Medium**

`CoreVesting.batchCreateVesting` divides by token amounts.

Recommendations:

Ensure that the token remainders are handled correctly.

```
uint256 tokensPerReceiver = params.totalTokens / params.receivers.length;
uint256 remainder = params.totalTokens % params.receivers.length;
if (remainder > 0) {
    // Handle the remainder
}
```

Vulnerabilities Info

Gas Ineffiency: Redundant reads and SStores in loop

Vulnerability severity: **Info**

Vulnerability probability: **Info**

`CoreVesting.releaseTokens` has inefficient reads and writes in a loop.

```
for (uint256 i = 0; i < vesting.cliffTimes.length; i++) {
    if (block.timestamp >= vesting.cliffTimes[i] && vesting.cliffPercents[i] > 0) {
        uint256 cliffTokens = (vesting.totalTokens * vesting.cliffPercents[i]) / 100;
        tokensToRelease += cliffTokens;
        vesting.cliffPercents[i] = 0;
    }
}
```

This recomputes a static result and writes an array.

Recommendations:

Precompute `vesting.cliffTokens`, then

```
struct Vesting {
    uint256 cliffTime;
    uint256 cliffTokens;
    bool cliffReleased;
}
if (!vesting.cliffReleased && block.timestamp >= vesting.cliffTime) {
    tokensToRelease += vesting.cliffTokens;
    vesting.cliffReleased = true;
}
```

This implementation will save ~90% gas

Disclaimer

Disclaimer

This report is governed by the Fidesium terms and conditions.

This report does not constitute an endorsement or disapproval of any project or team, nor does it reflect the economic value or potential of any related product or asset. It is not investment advice and should not be used as the basis for investment decisions. Instead, this report provides an assessment intended to improve code quality and mitigate risks inherent in cryptographic tokens and blockchain technology.

Fidesium does not guarantee the absence of bugs or vulnerabilities in the technology assessed, nor does it comment on the business practices, models, or regulatory compliance of its creators. All services, reports, and materials are provided "as is" and "as available," without warranties of any kind, including but not limited to merchantability, fitness for a particular purpose, or non-infringement.

Cryptographic assets and blockchain technologies are novel and carry inherent technical risks, uncertainties, and the possibility of unpredictable outcomes. Assessment results may contain inaccuracies or depend on third-party systems, and reliance on them is solely at the Customer’s risk.

Fidesium assumes no liability for content inaccuracies, personal injuries, property damages, or losses related to the use of its services, reports, or materials. Third-party components are provided "as is," and any warranties are strictly between the Customer and the third-party provider.

These services and materials are intended solely for the Customer's use and benefit. No third party or their representatives may claim rights to or rely on these services, reports, or materials under any circumstances.