SMART CONTRACTS REVIEW

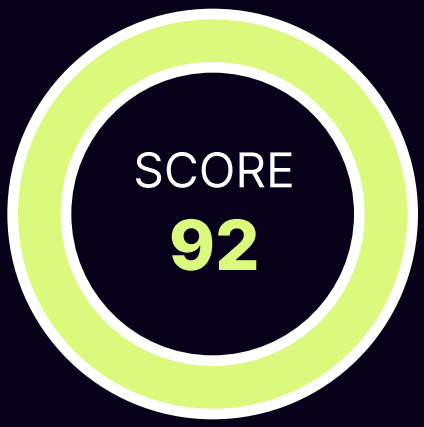
zokyo

March 28th 2024 | v. 1.0

Security Audit Score

PASS

Zokyo Security has concluded that these smart contracts passed a security audit.



ZOKYO AUDIT SCORING PARIBUS

1. Severity of Issues:
    - Critical: Direct, immediate risks to funds or the integrity of the contract. Typically, these would have a very high weight.
    - High: Important issues that can compromise the contract in certain scenarios.
    - Medium: Issues that might not pose immediate threats but represent significant deviations from best practices.
    - Low: Smaller issues that might not pose security risks but are still noteworthy.
    - Informational: Generally, observations or suggestions that don't point to vulnerabilities but can be improvements or best practices.
2. Test Coverage: The percentage of the codebase that's covered by tests. High test coverage often suggests thorough testing practices and can increase the score.
3. Code Quality: This is more subjective, but contracts that follow best practices, are well-commented, and show good organization might receive higher scores.
4. Documentation: Comprehensive and clear documentation might improve the score, as it shows thoroughness.
5. Consistency: Consistency in coding patterns, naming, etc., can also factor into the score.
6. Response to Identified Issues: Some audits might consider how quickly and effectively the team responds to identified issues.

HYPOTHETICAL SCORING CALCULATION:

Let's assume each issue has a weight:
- Critical: -30 points
- High: -20 points
- Medium: -10 points
- Low: -5 points
- Informational: -1 point

Starting with a perfect score of 100:
- 0 Critical issues: 0 points deducted
- 0 High issues: 0 points deducted
- 1 Medium issue: 1 acknowledged = - 4 points deducted
- 2 Low issues: 2 acknowledged = - 4 points deducted
- 1 Informational issue: 1 acknowledged = 0 points deducted

Thus, 100 - 4 - 4 = 92

TECHNICAL SUMMARY

This document outlines the overall security of the Paribus smart contracts evaluated by the Zokyo Security team.

The scope of this audit was to analyze and document the Paribus smart contracts codebase for quality, security, and correctness.

Contract Status

There were 0 critical issues found during the review. See Complete Analysis.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contracts but rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that can withstand the Ethereum network's fast-paced and rapidly changing environment, we recommend that the Paribus team put in place a bug bounty program to encourage further active analysis of the smart contracts.

Table of Contents

AUDITING STRATEGY AND TECHNIQUES APPLIED

The source code of the smart contract was taken from the Paribus repository:
Repo: https://github.com/Paribus/paribus-protocol/tree/develop/contracts/ParibusOracle

Last commit: b20a648033b8a6b4158c28b37b46be100f14416d

Within the scope of this audit, the team of auditors reviewed the following contract(s):

- ParibusOracleDelegate.sol
- ParibusOracleDelegator.sol
- ParibusOracleInterfaces.sol

During the audit, Zokyo Security ensured that the contract:

- Implements and adheres to the existing standards appropriately and effectively;
- The documentation and code comments match the logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices, efficiently using resources without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the most recent vulnerabilities;
- Meets best practices in code readability, etc.

Zokyo Security has followed best practices and industry-standard techniques to verify the implementation of Paribus smart contracts. To do so, the code was reviewed line by line by our smart contract developers, who documented even minor issues as they were discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

01 Due diligence in assessing the overall code quality of the codebase.

02 Cross-comparison with other, similar smart contracts by industry leaders.

03 Thorough manual review of the codebase line by line.

Executive Summary

The Zokyo team has performed a security audit of the provided codebase. The contracts submitted for auditing are well-crafted and organized. Detailed findings from the audit process are outlined in the "Complete Analysis" section.

STRUCTURE AND ORGANIZATION OF THE DOCUMENT

For the ease of navigation, the following sections are arranged from the most to the least critical ones. Issues are tagged as “Resolved” or “Unresolved” or “Acknowledged” depending on whether they have been fixed or addressed. Acknowledged means that the issue was sent to the Paribus team and the Paribus team is aware of it, but they have chosen to not solve it. The issues that are tagged as “Verified” contain unclear or suspicious functionality that either needs explanation from the Client or remains disregarded by the Client. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

FINDINGS SUMMARY

#	Title	Risk	Status
1	Centralization issues in ParibusOracleDelegate	Medium	Acknowledged
2	For loop over Dynamic Array	Low	Acknowledged
3	Missing sanity value checks	Low	Acknowledged
4	Solidity naming conventions not followed	Informational	Acknowledged

Centralization issues in ParibusOracleDelegate

The _setCollectionWhitelisted() function can be used by an owner to add or remove an NFT collection from whitelist anytime. This can lead to a malicious admin unfairly removing any NFT collection from the whitelist anytime.

Also _setUpdater() function can be used anytime by an admin to change the updater of the contract. This can have a strong impact on the contract as setCollectionPricesWei() and setPricesWei() can be called by an updater. And changing an updater to a compromised address can lead to setCollectionPricesWei() and setPricesWei() being exploited and set to unfair values.

Recommendation:
It is advised to use multisig wallets for the ownership of the contract, to allow more decentralization and prevent loss of private keys or malicious behaviours. Additionally it is advised to use a 3/5 or 2/3 configuration for the multisig wallets being used.

Comments: The team said that they will be considering adding more decentralization in future.

For loop over Dynamic Array

In ParibusOracleDelegate.sol, there is for loop over dynamic array in _setCollectionWhitelisted() function on line: 32 as follows:

```
for (uint i = 0; i < nfts.length; i++) {
    collectionWhitelist[nfts[i]] = whitelisted;
}
```

Similarly, there is for loop over dynamic array on line: 62 in setCollectionPricesWei() function:

```
for (uint i = 0; i < dataLen; i++) {
    TokenPrice storage tokenPrice = collectionPrices[tokenIds[i]];
    tokenPrice.priceWei = pricesWei[i];
    tokenPrice.updatedAt = getBlockTimestamp();
}
```

And there is for loop over dynamic array on line: 73 in setPricesWei() function:

```
for (uint i = 0; i < dataLen; i++) {
    TokenPrice storage tokenPrice = tokenPrices[nfts[i]][tokenIds[i]];
    tokenPrice.priceWei = pricesWei[i];
    tokenPrice.updatedAt = getBlockTimestamp();
}
```

This can lead to out of gas issues if length if the array being passed is too large.

Recommendation:

Although onlyOwner access control modifier has been used which prevents DDoS as well as access control issues, it is advised to keep a keen eye on the length of the array being passed so that the loop does not run out of gas while calling the these functions.

LOW-2 | ACKNOWLEDGED

Missing sanity value checks

In ParibusOracleDelegate.sol, there is missing sanity value checks for pricesWei calldata array parameter for both setCollectionPricesWei() and setPricesWei() functions. This can lead to incorrect values such as `0` value being assigned.

Recommendation:
It is advised to add sanity value checks for such as non-zero value checks, max value checks, etc. for values of pricesWei array, before being assigned to tokenPrice.priceWei.

INFORMATIONAL-1 | ACKNOWLEDGED

Solidity naming conventions not followed

In ParibusOracleDelegator.sol, the solidity naming conventions have not been followed for naming the function _setImplementation(). The underscore should be used for naming only internal functions in Solidity. But _setImplementation() is a public function and not an internal function.
Similarly, in the ParibusOracleDelegate.sol the naming convention has again been not followed. The function _setUpdater() uses an underscore even though it is not an internal function.

Recommendation:
It is advised to rename the functions according to the Solidity naming conventions for better code readability and remove the underscore from the names of-
1. _setImplementation() in ParibusOracleDelegator.sol and
2. _setUpdater() in ParibusOracleDelegate.sol

	ParibusOracleDelegate.sol ParibusOracleDelegator.sol ParibusOracleInterfaces.sol
Reentrance	Pass
Access Management Hierarchy	Pass
Arithmetic Over/Under Flows	Pass
Unexpected Ether	Pass
Delegatecall	Pass
Default Public Visibility	Pass
Hidden Malicious Code	Pass
Entropy Illusion (Lack of Randomness)	Pass
External Contract Referencing	Pass
Short Address/ Parameter Attack	Pass
Unchecked CALL Return Values	Pass
Race Conditions / Front Running	Pass
General Denial Of Service (DOS)	Pass
Uninitialized Storage Pointers	Pass
Floating Points and Precision	Pass
Tx.Origin Authentication	Pass
Signatures Replay	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass

We are grateful for the opportunity to work with the Paribus team.

The statements made in this document should not be interpreted as an investment or legal advice, nor should its authors be held accountable for the decisions made based on them.

Zokyo Security recommends the Paribus team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

zokyo