Audit Report

Neptune Updates

v1.0

January 9, 2024

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Cryptech Developments Ltd. to perform a security audit of Neptune Protocol.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/cryptechdev/neptune-protocol
Commit	`addc118b4b21be17ecdf2e1705dd7f208ed3946d`
Scope	In scope of this audit were all changes since our previous audit, which was performed at commit `4fe9176b31cf7a47f5bb4705c981861289ef24d9`, except the `leverage_trading` and `swap` features.
Fixes reviewed at commit	`8faed87e87cac24a67bd45f0de628ded25197a5f`

Repository	https://github.com/cryptechdev/neptune-auth
Commit	`8751d0bdaea482ae5574fbebfdcd1a264b29588a`
Scope	In scope of this audit were all changes since our previous audit, which was performed at commit `c8127eab528e77f9189988e81ee0485e261c408b`.
Fixes reviewed at commit	No fixes were submitted by the client.

Repository	https://github.com/cryptechdev/neptune-common
Commit	`8ef6906348ed5b14ba1aea183a252142fc91be1e`
Scope	In scope of this audit were all changes since our previous audit, which was performed at commit `efd313d353fafde910b9e03a9adc5c7449e3bed3`.
Fixes reviewed at commit	No fixes were submitted by the client.

The hashes of the compiled files at the above commits are:

`flash_loan_receiver.wasm`	`f2ec80fea4850cefdef3fbdb23befd0f8b42a810dfc75c8e37314c40fd511816`
`interest_model.wasm`	`ade251fe8445ce7f9813aa67926fba3d89e0911627b15d8167df2546ef3ccc2a`
`market.wasm`	`c0566526fb352eff6672bc9b9f0fa64ad77c1f91b8b58ec3fec7e5e3990b2090`
`price_oracle.wasm`	`a076e28d01a0944012949364b64ac19eb347c483d57e179a5860b5543ca4a6e5`
`querier.wasm`	`b876ce6e555e20c2ecb2e8b7f9c2451d762ce7defb7a4881b54f1a368bc9bf58`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Neptune Protocol is a decentralized lending protocol with a novel PID-transformed interest rate curve.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	Due to the novelty of the interest rate mechanism, the reliance on different off-chain components, and a large feature set (with features such as flash loans), the code base is non-trivial in several places.
Code readability and clarity	Medium-High	-
Level of documentation	Medium-High	-
Test coverage	Medium-High	86.29% test coverage

Summary of Findings

No	Description	Severity	Status
1	Price timestamp scale discrepancies could disable market operations for assets using the Ojo oracle	Major	Resolved
2	Price oracle contract allows owner and bots to overwrite prices submitted in the same block leading to race conditions	Minor	Acknowledged
3	No asset removal possible	Minor	Acknowledged
4	Missing address deduplication	Minor	Acknowledged
5	Oracle centralization issue	Minor	Acknowledged
6	Asset update function can be used for price updates	Informational	Acknowledged
7	“Migrate only if newer” pattern is not followed	Informational	Acknowledged
8	Contracts should implement a two step ownership transfer	Informational	Acknowledged
9	Outstanding ToDo comment in codebase	Informational	Acknowledged
10	No events emitted when withdrawing assets	Informational	Resolved
11	Automated asset withdrawal from the flash loan receiver might fail if too many assets are requested	Informational	Acknowledged

Detailed Findings

1. Price timestamp scale discrepancies could disable market operations for assets using the Ojo oracle

Severity: Major

In `contracts/price-oracle/src/query.rs:129-136`, when retrieving prices from Ojo oracles, the `ReferenceData` response returns `last_updated_base` and `last_updated_quote` timestamps in seconds, as specified in the documentation here.

However, the `query_ojo_price` function utilizes `Timestamp::from_nanos`, causing timestamps to be incorrectly scaled and stored, as they have not been converted to nanoseconds when storing the values.

Consequently, the `query_prices` function defined in `contracts/market/src/utility.rs:420-444`, will return a `MoneyMarketError::PriceTooOld` error leading to the impossibility of querying prices for assets that are using Ojo.

This would make Market contract messages like `ExecuteMsg::Borrow` and `ExecuteMsg::WithdrawCollateral` revert when targeting assets with an Ojo price oracle, potentially locking user funds.

Recommendation

We recommend aligning the time units consistently by converting `last_updated_base` and `last_updated_quote` prices to nanoseconds before storing them in the contract.

Status: Resolved

2. Price oracle contract allows owner and bots to overwrite prices submitted in the same block leading to race conditions

Severity: Minor

The price oracle contract allows `Owner` and `Bots` to publish asset prices using the `execute_update_prices` function defined in `contracts/price_oracle/src/contract.rs:56-77`.

However, since multiple `Bots` can submit their price in the same block and the oracle contract stores only one price for each asset, race conditions are possible where the latest transaction overwrites all the other submitted data.

Consequently, it might lead to users potentially fetching different prices within the same block, depending on the order of transactions which validators could manipulate to take profit.

Recommendation

We recommend aggregating prices from multiple bots rather than overwriting them. For example, the median price could be calculated per block to make the oracle more robust to compromised/malicious bots.

Status: Acknowledged

3. No asset removal possible

Severity: Minor

There is no functionality implemented to remove assets. If tokens turn malicious or regulators require the delisting of a certain asset, the lack of asset removal functionality might have a negative impact on the reputation of the protocol and affect uninformed users.

Recommendation

We recommend adding functionality to remove assets.

Status: Acknowledged

4. Missing address deduplication

Severity: Minor

In `packages/neptune-money-market/src/config.rs:121-137`, the `set_config` function allows the `Owner` to update the contract's configuration.

However, no deduplication is performed when storing addresses provided in `msg.vecs`.

Consequently, the same address may be stored multiple times, leading to inefficiencies when iterating the vector.

Recommendation

We recommend deduplicating addresses in vectors before storing them in the contract.

Status: Acknowledged

5. Oracle centralization issue

Severity: Minor

In `contracts/price-oracle/src/contract.rs:114-121`, when executing the `executing_update_prices` function, `Owner` and `Bots` are able to set arbitrary prices for `OracleAssetDetails::Regular` oracles.

This introduces a centralization concern, as bots and owners can set prices without any validation, price aggregation or Time-Weighted Average Price (TWAP) mechanism. Malicious bots can hence easily manipulate prices to their advantage.

Additionally, since there is no validation in place for the provided `price_info`, it allows setting a `price` of 0 or a `confidence` level of 0, leading to liquidations or undercollaterized loans.

We classify this issue as minor since the owner and bots are trusted entities.

Recommendation

We recommend implementing validation checks in the `executing_update_prices` process to ensure that the provided `price_info` is validated, preventing the submission of `prices` with values of 0 or `confidence` levels of 0.

Additionally, we recommend integrating a robust validation mechanism for price changes over time and incorporating price aggregation and Time-Weighted Average Price (TWAP) calculations.

Status: Acknowledged

6. Asset update function can be used for price updates

Severity: Informational

For updating prices, there is a dedicated function `executing_update_prices` in `contracts/price-oracle/src/contract.rs:114-121`. However, the function `execute_update_asset` in line `142` can also be used to update the prices of existing assets.

Because of this, it will not be possible to completely separate these two tasks: If there are in the future two roles where one is only allowed to add new assets (via `execute_update_asset`) and the other role is only allowed to update the prices of existing assets, the one that is allowed to add new assets can also update prices. Therefore, a complete separation of concerns cannot be achieved, violating the principle of least privilege.

Recommendation

We recommend disallowing the update of prices via `execute_update_asset`.

Status: Acknowledged

7. "Migrate only if newer" pattern is not followed

Severity: Informational

In

- `contracts/flash-loan-receiver/src/contract.rs:37-44`,
- `contracts/interest-model/src/contract.rs:48-55`,
- `contracts/market/src/contract.rs:43-50`, and
- `contracts/price-oracle/src/contract.rs:44-51`,

the contracts within the scope of this audit are currently migrated without regard of their version.

This can be improved by adding validation to ensure that the migration is only performed if the supplied version is newer.

Recommendation

We recommend following the migrate "only if newer" pattern defined in the CosmWasm documentation.

Status: Acknowledged

8. Contracts should implement a two step ownership transfer

Severity: Informational

The contracts within the scope of this audit allow the current owner to execute a one-step ownership transfer. While this is common practice, it presents a risk for the ownership of the contract to become lost if the owner transfers ownership to the incorrect address. A two-step ownership transfer will allow the current owner to propose a new owner, and then the account that is proposed as the new owner may call a function that will allow them to claim ownership and actually execute the config update.

Recommendation

We recommend implementing a two-step ownership transfer. The flow can be as follows:

1. The current owner proposes a new owner address that is validated and lowercased.
2. The new owner account claims ownership, which applies the configuration changes.

Status: Acknowledged

9. Outstanding TODO comment in codebase

Severity: Informational

In `contracts/market/src/execute.rs:1161` a comment says: "This summary is deprecated and will be removed in future versions."

Recommendation

We recommend resolving and removing deprecated code from the codebase.

Status: Acknowledged

10. No events emitted when withdrawing assets

Severity: Informational

In `contracts/flash-loan-receiver/src/contract.rs:79-94`, the `withdraw_assets` function does not emit any events. Bots and admins might require withdrawal information for further processing and will have to query the information otherwise separately.

Recommendation

We recommend emitting events when withdrawing assets.

Status: Resolved

11. Automated asset withdrawal from the flash loan receiver might fail if too many assets are requested

Severity: Informational

The iteration over assets in `contracts/flash-loan-receiver/src/contract.rs:87` might run out of gas if too many assets are requested at once. As a consequence, the assets will remain in the flash loan receiver. As the withdrawal is triggered by bots run by Neptune, this might cause inefficient calls consuming gas.

Recommendation

We recommend limiting the amount of assets that can be withdrawn at once.

Status: Acknowledged