HashCloak

Code Review and Security Assessment
For
Pyth Network

Delivery: June 10, 2024
Updated Delivery: Sept 3, 2024

Prepared For:
Daniel Chew | *Pyth Network*
Abhimanyu Bansal | *Pyth Network*
Wayne van Niekerk | *Fuel Labs*

Prepared by:
Ghilia Weldesselasie | *HashCloak Inc.*
Mikerah Quintyne-Collins | *HashCloak Inc.*

Table Of Contents

Executive Summary

Pyth Network engaged *HashCloak Inc.* for an audit of the Pyth Sway SDK contracts, a set of contracts for using real-time Pyth Network price data on the Fuel Network. Pyth Network is an oracle protocol that connects the owners of market data to applications on multiple blockchains, now including Fuel. Market data is provided by some of the biggest crypto exchanges and market makers in the world and propagated to other blockchains via Wormhole, a cross-chain messaging protocol.

The audit was done from May 16, 2024 to June 7, 2024. The relevant codebase was the pyth-crosschain repository, assessed in branch `main` at commit ae8fab12172a46ec1a73f5b134482c0f89677ba3.
The scope of the audit were all files in the following folders:

- `target_chains/fuel/contracts/pyth-contract`
- `target_chains/fuel/contracts/pyth-interface`

A review of the smart contracts was done on Sept 3, 2024 on the updated commit 1bad16e87b7fb02db51c5054cbd6d48edc114e9b.

During the audit, we familiarized ourselves with the inner workings of the protocol, the mechanics of Wormhole, and how messages get signed and passed on, as well as the necessary safety checks price oracle SDKs must perform (particularly other Pyth SDK implementations) to ensure applications can safely use prices provided by the Pyth Network.

Overall, we found the issues range from medium to informational.

Severity	Number of Findings
Critical	0
High	0
Medium	1
Low	2
Informational	2

Overview

Pyth Network is a network of trusted and reputable exchanges and market makers providing accurate real-time price data across 50+ blockchains. Pythnet runs as its own separate blockchain, that providers submit price data to, which then funnels those updates over to Wormhole to be distributed to other blockchains.

Pyth uses a new *Pull Oracle* design to allow *anyone* to update the price of an asset onchain as soon as it is available on Pythnet and signed by a quorum of Wormhole Guardians. This allows for more frequent updates and passes on the cost of doing so to whoever requests the update, instead of less frequent, periodic updates with costs borne by the oracle protocol.

Pyth has developed a suite of SDKs to integrate Pyth price data in different smart contract environments. The focus of this audit is the Sway implementation of the Pyth SDK on Fuel. These SDKs typically include utilities for querying the latest price and checking that it is current (i.e., not stale), updating the price and verifying that it originates from Pythnet and Wormhole, and supporting governance updates such as setting update fees or upgrading the contracts.

Methodology

Our audit methodology involved the following steps:

1. Go over the Pyth Network documentation provided by Pyth team.
2. Based on the documentation, we formulated an initial checklist that combines items for the most common Oracle integration bugs and our initial ideas on areas in which we believe there might be a cause for concern. The Merkle Tree utilities were also checked for common vulnerabilities.
3. Verify that issues that surfaced in previous audits of other SDK implementations were accounted for and patched in this Sway implementation.
4. Check that encoded messages from Pythnet cannot be replayed or misused, such as by passing an accumulator update into a batch attestation update function, or using governance messages as price updates.
5. As a final step, we utilized sway-analyzer, an off-the-shelf analysis tool, to surface any overlooked bugs. While no critical issues were surfaced, we did identify several code quality issues, such as unused imports, and the use of magic numbers.

Findings

PN-01: Guardians are assumed trusted and no input checks are done

Type: Medium
Files affected:

- `pyth-crosschain/target_chains/fuel/contracts/pyth-interface/src/data_structures/accumulator_update.sw`
- `pyth-crosschain/target_chains/fuel/contracts/pyth-interface/src/data_structures/batch_attestation_update.sw`
- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/src/main.sw`

Description: The Wormhole Guardians who sign price updates are trusted only to sign price updates from Pythnet, as such no safety checks are done on the payloads sent from Wormhole. An appropriate check could be done on `L120` in `accumulator_update.sw` to ensure that `price_feed.price.publish_time` is not set too far out in the future.

This is the current code on `L120-L123` without that added check:

```
if price_feed.price.publish_time > latest_publish_time {
    latest_price_feed.insert(price_feed.id, price_feed);
    updated_ids.push(price_feed.id);
}
```

Impact: If a quorum of Wormhole Guardians were to be compromised or act maliciously, they could sign a price update far out into the future (e.g., `u64::max()`) and completely prevent any further updates, as no new price could have a later `publish_time`.

Suggestion: An additional storage variable `max_publish_time_gap` can be defined which can then be used to also check that `price_feed.price.publishing_time - latest_publish_time <= max_publish_time_gap`.

Status: The Pyth development team has rectified this issue by adding a `require` statement within `PriceFeed.parse_messages`' implementation here.

PN-02: Absolute difference function allows price updates from the future

Type: Low
Files affected:

- `pyth-crosschain/target_chains/fuel/contracts/pyth-interface/src/utils.sw`
- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/src/main.sw`

Description:
The difference function in utils.sw is an absolute difference function, which always returns a non-negative result. In this context, `timestamp()` is *always* expected to be greater than `price.publish_time`. If `price.publish_time` were greater than `timestamp()` that would mean the price update was somehow from the future, this is invalid behavior and should revert the execution. However, an absolute function won't revert and instead will return `price.publish_time - timestamp()` in the case that the price was published in the future `(price.publish_time > timestamp())`.

```
fn price_no_older_than(time_period: u64, price_feed_id: PriceFeedId)
-> Price {
   let price = price_unsafe(price_feed_id);
   require(
--->   difference(timestamp(), price.publish_time) <= time_period,
       PythError::OutdatedPrice,
   );

   price
}
```

Suggestion: If accepting price updates from the future is not intended, then an *order dependent* difference function should be used. The absolute difference is order *independent* (mathematically, this is called the symmetry property). While this is a minor informational issue, it is important to note that price updates must be provided by trusted data providers and signed by a quorum of Wormhole guardians. Therefore, we can reasonably assume that invalid updates are unlikely to be signed.

Status: The Pyth development team has removed the difference function from `utils.sw` here.

PN-03: Public Struct fields must be declared public

Type: Low
Files affected:

- `pyth-crosschain/target_chains/fuel/contracts/pyth-interface/src/data_structures`

Description and suggestion: Public fields in struct must be declared public with the keyword `pub` to be used outside the module.
Status: The Pyth team already patched this in commit 3513eed.

PN-04: Address TODOs

Type: Informational

Description: There are a few open TODOs throughout the codebase, such as:

- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/src/events.sw:L14`
- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/data_structures/update_type.sw:L18`
- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/src/main.sw:`
  - `L388`
  - `L404`
  - `L417`
  - `L428`
  - `L830`

Suggestion: These should all be addressed before deployment of the contracts.

Status: The current list of TODOs as of commit 1bad16e87b7fb02db51c5054cbd6d48edc114e9b pertains to either features that need to be available within the Sway language, or to upgradability of the Pyth Sway contracts.

PN-05: Define constants, avoid using magic numbers

Type: Informational

Files affected:

- `pyth-crosschain/target_chains/fuel/contracts/pyth-contract/src/`

Description: Multiple so-called magic numbers are undefined and not well documented in the code. These can be surfaced with the command: `sway-analyzer --directory pyth-contract/ --detectors magic_number`

Suggestion: Constants can be defined instead, such as `const UPDATE_TYPE_SIZE = 8;`.