ClarityAlliance

GRANITE-PYTH (UPGRADE) SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA)

AUGUST 15TH, 2025

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

1. About Clarity Alliance

Clarity Alliance is a team of expert whitehat hackers specialising in securing protocols on Stacks.

They have disclosed vulnerabilities that have saved millions in live TVL and conducted thorough reviews for some of the largest projects across the Stacks ecosystem.

Learn more about Clarity Alliance at clarityalliance.org.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

2. Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Clarity Alliance to perform a security assessment.

This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Clarity Alliance’s position is that each company and individual are responsible for their own due diligence and continuous security. Clarity Alliance’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Clarity Alliance are subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis.

Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third parties. Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Clarity Alliance does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

3. Introduction

A time-boxed security review of the Pyth Oracle Client implementation for the Stacks blockchain, where Clarity Alliance reviewed the scope and provided insights on improving the protocol.

4. About Pyth Oracle

Pyth Network is an oracle that publishes financial market data to multiple blockchains. The market data is contributed by over 80 first-party publishers, including some of the biggest exchanges and market-making firms in the world. Pyth offers price feeds for several asset classes, including US equities, commodities, and cryptocurrencies. Each price feed publishes a robust aggregate of publisher prices that updates multiple times per second. Price feeds are available on multiple blockchains and can be used in off-chain applications.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review

Granite-Pyth
(Upgrade)

5. Risk Classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1 Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2 Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

6. Security Assessment Summary

Scope

The following contracts were in the scope of the security review:

- `contracts/pyth-governance-v2.clar`
- `contracts/pyth-pnau-decoder-v2.clar`
- `contracts/wormhole/wormhole-core-v3.clar`

Initial Commit Reviewed:
5a630a4a6088e84e28a5a99562e9249d7a2517be

Final Commit After Remediations:
1f3a7a93b76e33f648e3c94ba292bdee2f91c58f

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Alin Barbatei (ABA) engaged with - to review Pyth Oracle. In this period of time a total of **16** issues were uncovered.

Protocol Summary

Protocol Name	Pyth Oracle
Date	August 15th, 2025

Findings Count

Severity	Amount
High	1
Low	1
QA	14
Total Findings	**16**

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
  5.1. Impact 5
  5.2. Likelihood 5
  5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
  [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
  [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
  [QA-01] Redundant Return Data When Parsing PTGM 11
  [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
  [QA-03] Remove Debug Remnants 13
  [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
  [QA-05] Redundant Merkle Ring Size Read Operation 15
  [QA-06] Redundant Return Data When Parsing PNAU Header 16
  [QA-07] Optimization of Parsing and Verifying Price Updates 17
  [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
  [QA-09] Redundant Reading Operations When Parsing VAAs 21
  [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
  [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
  [QA-12] Improve Codebase Comments 25
  [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
  [QA-14] Increment Contract Versions 28

Summary of Findings

ID	Title	Severity	Status
[H-01]	Incorrect Processing of Price Batch Update Feeds 5 and 6	High	Resolved
[L-01]	Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches	Low	Resolved
[QA-01]	Redundant Return Data When Parsing PTGM	QA	Resolved
[QA-02]	Reuse Offset Variable in Governance Data Source Parsing	QA	Resolved
[QA-03]	Remove Debug Remnants	QA	Resolved
[QA-04]	Redundant Return Data When Parsing Merkle Root Data	QA	Resolved
[QA-05]	Redundant Merkle Ring Size Read Operation	QA	Resolved
[QA-06]	Redundant Return Data When Parsing PNAU Header	QA	Resolved
[QA-07]	Optimization of Parsing and Verifying Price Updates	QA	Resolved
[QA-08]	Wormhole VAA Parsing Can Be Slightly Improved	QA	Resolved
[QA-09]	Redundant Reading Operations When Parsing VAAs	QA	Resolved
[QA-10]	Reuse Already Declared Local Variables When Recovering Public Key	QA	Resolved
[QA-11]	Some Buffer Manipulation Function Calls Can Be Inlined	QA	Resolved
[QA-12]	Improve Codebase Comments	QA	Resolved
[QA-13]	Miscellaneous Codebase Improvements for Reducing Runtime Costs	QA	Resolved
[QA-14]	Increment Contract Versions	QA	Resolved

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
   5.1. Impact 5
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
   [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
   [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
   [QA-01] Redundant Return Data When Parsing PTGM 11
   [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
   [QA-03] Remove Debug Remnants 13
   [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
   [QA-05] Redundant Merkle Ring Size Read Operation 15
   [QA-06] Redundant Return Data When Parsing PNAU Header 16
   [QA-07] Optimization of Parsing and Verifying Price Updates 17
   [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
   [QA-09] Redundant Reading Operations When Parsing VAAs 21
   [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
   [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
   [QA-12] Improve Codebase Comments 25
   [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
   [QA-14] Increment Contract Versions 28

8. Findings

8.1. High Findings

[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6

Description

In the current implementation for updating price feeds, the caller can provide up to 6 feeds. However, the logic in `pyth-pnau-decoder-v2::parse-price-info-and-proof` incorrectly uses the same offset value when reading the 4th, 5th, and 6th feed updates. As a result, the 5th and 6th updates are processed incorrectly and mirror the 4th update.

In the code, the `update5` and `update6` feeds use the same offset as `update4`:

```
(update4 (unwrap! (read-and-verify-update bytes (+ (message-length update1)
  (message-length update2) (message-length update3) offset)) (ok (list update1 update2 update3
(update5 (unwrap! (read-and-verify-update bytes (+ (message-length update1)
  (message-length update2) (message-length update3) offset)) (ok (list update1 update2 update3
(update6 (unwrap! (read-and-verify-update bytes (+ (message-length update1)
  (message-length update2) (message-length update3) offset)) (ok (list update1 update2 update3
```

This leads to any price feed update involving 5 or 6 feeds incorrectly updating only 4 feeds without returning an error. Integrating systems may inadvertently use incorrect prices if they rely on `pyth-oracle-v3: : verify-and-update-price-feeds` to update and retrieve the latest prices, assuming the input asset price feed order matches the output price updates index.

Additionally, this bug causes any update with more than 3 feeds to automatically generate 6 feeds, as feeds 5 and 6 are duplicates of feed 4.

Recommendation

Adjust the `update5` variable to begin parsing at an offset that includes the length of `update4` (`(message-length update4)`). Similarly, modify the `update6` variable to include the lengths of both `update4` and `update5` (`(message-length update4) (message-length updates)`).

ClarityAlliance
Security Review

Granite-Pyth
(Upgrade)

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

8.2. Low Findings

[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches

Description

In the `pyth-pnau-decoder-v2` decoder contract, when parsing and verifying feed updates within the `parse-and-verify-prices-updates` function, there are two critical checks:

The most important check ensures that the number of feeds to be updated matches the number of feed data entries provided for the update:

```
(asserts! (is-eq num-updates (len updates)) ERR_INCORRECT_AUWV_PAYLOAD)
```

The second check ensures that the parsed buffer does not have any overlay, which could indicate a potentially corrupt format:

```
;;
   Overlay check; 1 is added because 1 byte is used to store “cursor-num updates”
(asserts! (is-eq (+ (fold sum-message-length updates u0) u1)
  (len bytes)) ERR_OVERLAY_PRESENT)
```

Both of these checks have been moved into an `if` statement and are only applied if there are fewer than 3 or exactly 6 updates:

```
(if (or (<= num-updates u3) (is-eq num-updates u6))
```

By placing the checks within the `if` clause, there is a risk that incorrect or malformed buffers may be accepted when the number of updates requested does not match the actual data provided. Although this scenario is rare and would require the Pyth trusted API to generate such an input incorrectly, these checks should be applicable in all cases.

Recommendation

Relocate the two checks from the `(if (or (<= num-updates u3) (is-eq num-updates u6))` block back into the main body of the `parse-and-verify-prices-updates` function.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

8.3. QA Findings

[QA-01] Redundant Return Data When Parsing PTGM

Description

In the `pyth-governance-v2` contract, whenever Pyth governance issues changes, the payload is read and validated via the `parse-and-verify-ptgm` function.

The function returns `actions`, `target-chain-id`, `module`, `cursor` and `body`:

```
(ok {
  action: action,
  target-chain-id: target-chain-id,
  module: module,
  cursor: target-chain-id,
  body: body
})))
```

The private `parse-and-verify-ptgm` function is called from nine different locations. In all these instances, only `action` and `body` are utilized:

The `target-chain-id` can be removed since its sole purpose was to ensure it was a Stacks governance message, a check already performed within the `parse-and-verify-ptgm` function. The same applies to `module`, which is also checked within the function and is not used elsewhere.

The `cursor` value is a leftover from previous versions of the codebase, where a cursor-like object was used for `buf` parsing. This has changed, and currently, the value is both unused and serves no purpose, being randomly assigned to the target chain id.

Removing these three return entries from the tuple would simplify the code and slightly reduce execution costs.

Recommendation

Remove the `target-chain-id`, `module`, and `cursor` entries from the `pyth-governance-v2:: parse-and-verify-ptgm` return value.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
  5.1. Impact 5
  5.2. Likelihood 5
  5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
  [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
  [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
  [QA-01] Redundant Return Data When Parsing PTGM 11
  [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
  [QA-03] Remove Debug Remnants 13
  [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
  [QA-05] Redundant Merkle Ring Size Read Operation 15
  [QA-06] Redundant Return Data When Parsing PNAU Header 16
  [QA-07] Optimization of Parsing and Verifying Price Updates 17
  [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
  [QA-09] Redundant Reading Operations When Parsing VAAs 21
  [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
  [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
  [QA-12] Improve Codebase Comments 25
  [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
  [QA-14] Increment Contract Versions 28

[QA-02] Reuse Offset Variable in Governance Data Source Parsing

Description

In the `pyth-governance-v2: : parse-data-source` function, the `offset` variable is initialized as `(get index (get cursor acc))` to simplify usage and reduce fees.

However, within the same code namespace, the operation `(get index (get cursor acc))` is redundantly repeated twice more when returning the `index` and `next-update-index` variables.

```
cursor: {
  index: (+ (get index (get cursor acc)) u1),
  next-update-index: (+ (get index (get cursor acc)) SIZE_OF_EMITTER_DATA),
},
```

This redundancy unnecessarily increases execution costs and decreases readability.

Recommendation

Utilize the `offset` variable in the `pyth-governance-v2:: parse-data-source` function when returning the `index` and `next-update-index` variables.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

[QA-03] Remove Debug Remnants

Description

Within the `parse-and-verify-prices-updates` function of the `pyth-pnau-decoder-v2` contract, a `print` statement containing the action `test` remains as a debug remnant.

Recommendation

Eliminate this statement to reduce execution costs and ensure consistency across the codebase.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
    5.1. Impact 5
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
  [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
  [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
  [QA-01] Redundant Return Data When Parsing PTGM 11
  [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
  [QA-03] Remove Debug Remnants 13
  [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
  [QA-05] Redundant Merkle Ring Size Read Operation 15
  [QA-06] Redundant Return Data When Parsing PNAU Header 16
  [QA-07] Optimization of Parsing and Verifying Price Updates 17
  [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
  [QA-09] Redundant Reading Operations When Parsing VAAs 21
  [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
  [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
  [QA-12] Improve Codebase Comments 25
  [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
  [QA-14] Increment Contract Versions 28

[QA-04] Redundant Return Data When Parsing Merkle Root Data

Description

In the decoder contract, the function `parse-merkle-root-data-from-vaa-payload` is used to parse the Merkle root. The function returns a tuple containing the following elements:

```
(ok {
  value: {
    merkle-root-slot: merkle-root-slot,
    merkle-root-ring-size: merkle-root-ring-size,
    merkle-root-hash: merkle-root-hash,
    payload-type: payload-type
  },
  next: merkle-root-hash
 }))
```

However, only the `value.merkle-root-hash` is utilized, despite all elements being returned. The code snippet below demonstrates this usage:

```
(cursor-merkle-root-data (try! (parse-merkle-root-data-from-vaa-payload
  (get payload vaa))))
(decoded-prices-updates (try! (parse-and-verify-prices-updates
  (slice pnau-bytes (+ offset u2 pnau-vaa-size) none) (get merkle-root-hash (get value cursor-
```

This results in unnecessary code complexity and a slight increase in execution fees.

Recommendation

Modify the `pyth-pnau-decoder-v2:: parse-merkle-root-data-from-vaa-payload` function to return only the `merkle-root-hash` directly, rather than a nested tuple.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
    5.1. Impact 5
    5.2. Likelihood 5
    5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
  [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
  [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
  [QA-01] Redundant Return Data When Parsing PTGM 11
  [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
  [QA-03] Remove Debug Remnants 13
  [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
  [QA-05] Redundant Merkle Ring Size Read Operation 15
  [QA-06] Redundant Return Data When Parsing PNAU Header 16
  [QA-07] Optimization of Parsing and Verifying Price Updates 17
  [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
  [QA-09] Redundant Reading Operations When Parsing VAAs 21
  [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
  [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
  [QA-12] Improve Codebase Comments 25
  [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
  [QA-14] Increment Contract Versions 28

[QA-05] Redundant Merkle Ring Size Read Operation

Description

In the decoder, the function `parse-merkle-root-data-from-vaa-payload` is responsible for parsing VAA for Merkle data. Within this function, the Merkle root ring size is read as follows:

```
(merkle-root-ring-size (unwrap!
  (read-uint-32 payload-vaa-bytes u13) ERR_INVALID_AUWV))
```

This read operation is redundant because the ring size is not utilized by Pyth. This behavior aligns with the EVM implementation of Pyth:

```
// This field is not used
// uint32 ringSize = UnsafeBytesLib.toUint32(encodedPayload, payloadoffset);
payloadOffset += 4;
```

Previously, this was retained due to the parsing logic using cursor-relative indexed processing for buffers/byte arrays. However, with the new absolute-offset implementation, reading the `merkle-root-ring-size` variable is unnecessary.

This results in an unnecessary real-time execution cost for each decode call.

Recommendation

Remove the `merkle-root-ring-size` variable and its read operation entirely from the `pyth-pnau-decoder-v2::parse-merkle-root-data-from-vaa-payload` function. Additionally, include a comment to clarify that this removal is intentional, ensuring readers understand it is not an oversight but a deliberate action.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

[QA-06] Redundant Return Data When Parsing PNAU Header

Description

In the decoder contract, when invoking `parse-pnau-header` to parse the PNAU header, the function returns a tuple containing the following elements:

```
(ok {
  value: {
    magic: magic,
    version-major: version-major,
    version-minor: version-minor,
    header-trailing-size: header-trailing-size,
    proof-type: proof-type
  },
  pos: (+ header-trailing-size u8)
}))
```

However, only the `pos` entry is utilized subsequently, despite all elements being returned:

```
(let ((pnau-header (try! (parse-pnau-header pnau-bytes)))
      (offset (get pos pnau-header))
```

This results in unnecessary code complexity and a slight increase in execution fees.

Recommendation

Modify the `pyth-pnau-decoder-v2:: parse-pnau-header` function to return only the required size: `(+ header-trailing-size u8)`, instead of a nested tuple.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

[QA-07] Optimization of Parsing and Verifying Price Updates

Description

The `pyth-pnau-decoder-v2` contract currently incurs high runtime fees. Despite previous optimizations, the `parse-and-verify-prices-updates` and `parse-price-info-and-proof` functions can be further optimized. This can be achieved by eliminating the use of the `message-length` and `sum-message-length` functions and directly calculating the size of updates during processing.

The proposed optimization includes:

- Adding an `update-size` entry in the return of the `read-and-verify-update` function, which directly holds the size of the currently parsed feed.

```
update-size: (+ u3 message-size (* MERKLE_PROOF_HASH_SIZE proof-size))
```

Additionally, further runtime optimization can be achieved by storing `(* MERKLE_PROOF _HASH_SIZE proof-size)` in a variable, as `proof-bytes` also uses this calculation:

```
@@ -228,10 +193,8 @@
    (ema-price (try! (read-int-64 bytes (+ offset u71))))
    (ema-conf (try! (read-uint-64 bytes (+ offset u79))))
    (proof-size (try! (read-uint-8 bytes (+ offset u2 message-size))))
-   (proof-bytes (default-to 0x (slice? bytes
-     (+ offset u3 message-size)
-     (+ offset u3 message-size (* MERKLE_PROOF_HASH_SIZE proof-size))
-   )))
+   (proof-length (* MERKLE_PROOF_HASH_SIZE proof-size))
+   (proof-bytes (default-to 0x (slice? bytes (+ offset u3 message-size)
+ (+ offset u3 message-size proof-length))))
    (leaf-bytes (default-to 0x (slice? bytes (+ offset u2)
      (+ offset u2 message-size))))
    (proof (get result (fold parse-proof proof-bytes {
          result: (list),
@@ -254,7 +217,8 @@
    ema-price: ema-price,
    ema-conf: ema-conf,
    proof: proof,
-   leaf-bytes: (unwrap-panic (as-max-len? leaf-bytes u255))
+   leaf-bytes: (unwrap-panic (as-max-len? leaf-bytes u255)),
+   update-size: (+ u3 message-size proof-length)
  })
))
```

- In the `parse-price-info-and-proof` function, add the size to the offset and increment the offset with each call. Modify the return type to be a tuple with the cumulative offset for use in the overlay check:

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

```
(define-private (parse-price-info-and-proof (bytes (buff 8192)))
  (let (
     (offset u1)
    (update1 (try! (read-and-verify-update bytes offset)))
-   (update2 (unwrap! (read-and-verify-update bytes (+
- (message-length update1) offset)) (ok (list update1))))
-   (update3 (unwrap! (read-and-verify-update bytes (+
- (message-length update1) (message-length update2) offset)) (ok (list update1 update2))))
-   (update4 (unwrap! (read-and-verify-update bytes (+
- (message-length update1) (message-length update2) (message-length update3) offset)) (ok (lis
-   (update5 (unwrap! (read-and-verify-update bytes (+
- (message-length update1) (message-length update2) (message-length update3) offset)) (ok (lis
-   (update6 (unwrap! (read-and-verify-update bytes (+
- (message-length update1) (message-length update2) (message-length update3) offset)) (ok (lis
+   (offset-1 (+ offset (get update-size update1)))
+   (update2 (unwrap! (read-and-verify-update bytes offset-1)
+ (ok { offset: offset-1, entries: (list update1)})))
+   (offset-2 (+ offset-1 (get update-size update2)))
+   (update3 (unwrap! (read-and-verify-update bytes offset-2)
+ (ok { offset: offset-2, entries: (list update1 update2)})))
+   (offset-3 (+ offset-2 (get update-size update3)))
+   (update4 (unwrap! (read-and-verify-update bytes offset-3)
+ (ok { offset: offset-3, entries: (list update1 update2 update3)})))
+   (offset-4 (+ offset-3 (get update-size update4)))
+   (update5 (unwrap! (read-and-verify-update bytes offset-4)
+ (ok { offset: offset-4, entries: (list update1 update2 update3 update4)})))
+   (offset-5 (+ offset-4 (get update-size update5)))
+   (update6 (unwrap! (read-and-verify-update bytes offset-5)
+ (ok { offset: offset-5, entries: (list update1 update2 update3 update4 update5)})))
   )
- (ok (list update1 update2 update3 update4 update5 update6))
+ (ok { offset: (+ offset-5 (get update-size update6)), entries:
+ (list update1 update2 update3 update4 update5 update6)})
  )
)
```

- In the `parse-and-verify-prices-updates`, retrieve the offset and include it in the overlay check.

```
(define-private (parse-and-verify-prices-updates (bytes (buff 8192))
  (merkle-root-hash (buff 20)))
  (let ((num-updates (try! (read-uint-8 bytes u0)))
        (max-updates-check (asserts!
          (<= num-updates MAXIMUM_UPDATES) ERR_MAXIMUM_UPDATES))
-       (updates (try! (parse-price-info-and-proof bytes)))
+       (update-data (try! (parse-price-info-and-proof bytes)))
+       (updates (get entries update-data))
        (merkle-proof-checks-success (get result
          (fold check-merkle-proof updates {
          result: true,
          merkle-root-hash: merkle-root-hash
        }))))
    (asserts! merkle-proof-checks-success ERR_MERKLE_ROOT_MISMATCH)
    (asserts! (is-eq num-updates (len updates)) ERR_INCORRECT_AUWV_PAYLOAD)
-   (asserts! (is-eq (+ (fold sum-message-length updates u0) u1)
- (len bytes)) ERR_OVERLAY_PRESENT)
+   (asserts! (is-eq (get offset update-data) (len bytes)) ERR_OVERLAY_PRESENT)
    (ok updates)
```

- The `message-length` and `sum-message-length` functions can be removed, and slight modifications are needed when passing the tuple due to the newly added `update-size` entry.

This optimization reduces both `read_length` and `runtime` costs as follows:

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

Metric	Before	After	Reduction
read_length	204162	203615	547 (0.27%)
runtime	5878957	5820855	58102 (0.99%)

Recommendation

Implement the suggested optimization in the codebase to reduce execution costs. .

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

[QA-08] Wormhole VAA Parsing Can Be Slightly Improved

Description

In the `wormhole-core-v3:: parse-vaa` function, there are several minor improvements that can enhance uniformity:

1. The term `singnatures-offset` contains a typo and should be corrected to `signatures-offset`.
2. Introduce a constant, such as `SIGNATURE_DATA_SIZE`, for the `u66` value, which semantically represents `(‹guardian_id_1byte | signature _65bytes)`.
3. Use `signatures-len` instead of `(len signatures)` when slicing the `vaa-body-hash-list`.

Recommendation

Implement the suggested changes in the codebase.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
 5.1. Impact 5
 5.2. Likelihood 5
 5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
 [H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
 [L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
 [QA-01] Redundant Return Data When Parsing PTGM 11
 [QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
 [QA-03] Remove Debug Remnants 13
 [QA-04] Redundant Return Data When Parsing Merkle Root Data 14
 [QA-05] Redundant Merkle Ring Size Read Operation 15
 [QA-06] Redundant Return Data When Parsing PNAU Header 16
 [QA-07] Optimization of Parsing and Verifying Price Updates 17
 [QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
 [QA-09] Redundant Reading Operations When Parsing VAAs 21
 [QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
 [QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
 [QA-12] Improve Codebase Comments 25
 [QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
 [QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review

Granite-Pyth
(Upgrade)

[QA-09] Redundant Reading Operations When Parsing VAAs

Description

In the wormhole VAA decoder contract, the `parse-vaa` function returns a tuple containing the following elements:

```
(ok {
vaa: {
      version: version,
      guardian-set-id: guardian-set-id,
      signatures-len: signatures-len,
      signatures: signatures,
      timestamp: timestamp,
      nonce: nonce,
      emitter-chain: emitter-chain,
      emitter-address: emitter-address,
      sequence: sequence,
      consistency-level: consistency-level,
      payload: payload,
    },
    recovered-public-keys: public-keys-results,
}))
```

Among the returned data, certain elements such as `timestamp`, `nonce` and `consistency-level` are never utilized, even within the `parse-vaa` function itself. These elements are redundant and contribute to increased execution costs during each VAA parsing:

```
(timestamp (unwrap!
  (read-uint-32 vaa-bytes singnatures-offset) ERR_VAA_PARSING_TIMESTAMP))
;; ... code ...
(nonce (unwrap! (read-uint-32 vaa-bytes
  (+ singnatures-offset u4)) ERR_VAA_PARSING_NONCE))
(consistency-level (unwrap! (read-uint-8 vaa-bytes
  (+ singnatures-offset u50)) ERR_VAA_PARSING_CONSISTENCY_LEVEL))
```

Additionally, while `signatures-len` and `signature` are used within the `parse-vaa` function, they are not utilized outside of it. Therefore, these elements, along with `timestamp`, `nonce`, `consistency-level`, can be removed from the return tuple.

Eliminating these unnecessary operations reduces code complexity and execution fees significantly. For instance, parsing a single feed update VAA shows the following improvements:

Metric	Before	After	Reduction
read_length	204162	204054	108 (0.05%)
runtime	5696215	5878957	182742 (3.11%)

Recommendation

In the `wormhole-core-v3: :parse-vaa` function, comment out the `timestamp`, `nonce`, and `consistency-level` entries, and include a note explaining their intentional exclusion. Remove these entries from the return tuple, along with `signatures-len` and `signatures` which are not used. Additionally update `wormhole-traits-vi` to reflect these changes.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review

Granite-Pyth
(Upgrade)

[QA-10] Reuse Already Declared Local Variables When Recovering Public Key

Description

In the `wormhole-core-v3:: recover-public-key` function, there are two local variables, `signature` and `guardian-id`, which are declared and used only once:

```
(signature (get signature entry))
(guardian-id (get guardian-id entry))
```

However, there are two additional instances where, instead of utilizing these variables, the `get` operation is performed again:

```
(let ((recovered-compressed-public-key (unwrap-panic
  (secp256k1-recover? message-hash (get signature entry)))))
```

Recommendation

In the `wormhole-core-v3: recover-public-key` function, reuse the `signature` and `guardian-id` variables.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
   5.1. Impact 5
   5.2. Likelihood 5
   5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review

Granite-Pyth
(Upgrade)

[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined

Description

Throughout the codebase, there are instances where function calls to buffer manipulation functions can be removed or simplified.

In the `wormhole-core-v3` contract:

1. The `read-buff-8192-max` function can be eliminated, as it is called only once from `parse-and-verify-guardians-set` Since it is invoked with a length, it can be directly replaced with a `read-buff` call:

```
- (guardians-bytes (unwrap! (read-buff-8192-max bytes u40 (some
- (* guardians-count GUARDIAN_ETH_ADDRESS_SIZE))) ERR_GSU_PARSING_GUARDIANS_
  BYTES))
+ (guardians-bytes (unwrap! (read-buff bytes u40
+ (* guardians-count GUARDIAN_ETH_ADDRESS_SIZE)) ERR_GSU_PARSING_GUARDIANS_BYTES))
```

2. The `read-buff-65` function can be removed, as it is called only once from `read-one-signature` and can be inlined:

```
 - signature: (unwrap-panic (read-buff-65 input u1))
 + signature: (unwrap-panic (as-max-len? (unwrap-panic
 + (slice? input u1 u66)) u65))
```

In the `pyth-governance-v2` contract:

1. The `read-buff-8192-max` function can be removed, as it is called only once from `parse-and-verify-ptgm` . Since it is called without a length, it can be directly replaced with a native `slice` call:

```
- (body (unwrap! (read-buff-8192-max ptgm-bytes u8 none) ERR_INVALID_PTGM))
+ (body (unwrap! (slice? ptgm-bytes u8 (len ptgm-bytes)) ERR_INVALID_PTGM))
```

In the `pyth-pnau-decoder-v2` contract:

1. The `read-buff-8192-max` function can be removed, as it is called only once from `decode-pnau-price-update` . Since it is called with a length, it can be directly replaced with a `read-buff` call:

```
 - (pnau-vaa (try! (read-buff-8192-max pnau-bytes (+ offset u2)
 - (some pnau-vaa-size))))
 + (pnau-vaa (try! (read-buff pnau-bytes (+ offset u2) pnau-vaa-size)))
```

2. The `slice` function can be removed, as it is used only once and can be replaced with an inline `slice?` call:

CONTENTS
1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

```
;; Invalid PNAU buffer, shorter than required
(define-constant ERR_INVALID_PNAU_BYTES (err u2404))
;; ... code ...
(encoded-price-updates (unwrap! (slice? pnau-bytes (+ offset u2 pnau-vaa-size)
  (len pnau-bytes)) ERR_INVALID_PNAU_BYTES))
(decoded-prices-updates (try!
  (parse-and-verify-prices-updates encoded-price-updates (get merkle-root-hash (get value curs
```

The unnecessary code complexity and increased fees from these operations are significant. By removing them, we achieve a reduction in both runtime execution costs and read length. For example, when parsing a 1 feed update VAA:

Metric	Before	After	Reduction
read_length	204162	201943	2219 (1.09%)
runtime	5878957	5516170	362787 (6.17%)

Recommendation

Implement all the mentioned optimizations in the codebase.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review
Granite-Pyth (Upgrade)

[QA-12] Improve Codebase Comments

Description

There are two instances of incorrect comments in the codebase:

1. In the `pyth-pnau-decoder-v2` contract, at line 33, the comment `;; Merkle root mismatch` is incorrect. It should be updated to `Incorrect AUWV message`.
2. In the `wormhole-core-v3` contract, at line 238, the word `atleast` contains a typo and should be corrected to `at least`.
3. The redundant comments `;; Good to go!` should be removed from the `wormhole-core-v3` contract at line 207 and line 380.

Recommendation

Update the comments as specified above.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

ClarityAlliance
Security Review

Granite-Pyth (Upgrade)

[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs

Description

Several minor changes can be implemented to slightly reduce codebase costs:

1. Modify all `read-int-*` functions to operate without the `let` block:

Example change:

From:

```
(define-private (read-int-32 (bytes (buff 8192)) (pos uint))
    (let ((cursor-bytes (try! (read-buff bytes pos u4))))
        (ok (bit-shift-right (bit-shift-left (buff-to-int-be (unwrap-panic
          (as-max-len? cursor-bytes u4))) u96) u96))))
```

To:

```
  (define-private (read-int-32 (bytes (buff 8192)) (pos uint))
    (ok (bit-shift-right (bit-shift-left (buff-to-int-be (unwrap-panic
      (as-max-len? (try! (read-buff bytes pos u4)) u4))) u96) u96)))
```

2. Cache values to reduce runtime costs:

- In `wormhole-core-v3::parse-and-verify-vaa` , the `(get vaa message)` value is accessed three times. Store it in a `vaa` variable and reuse it.
- In `wormhole-core-v3::parse-and-verify-guardians-set` , the `(* guardians-count GUARDIAN_ETH_ADDRESS_SIZE)` calculation is performed twice. Store it in a `guardians-bytes-size` variable and reuse it.
- In `pyth-pnau-decoder-v2::parse-proof` , within the `else` branch, store `(get index (get cursor acc))` and `(get next-update-index (get cursor acc))` in variables and reuse them.

3. Merge `decode-and-verify-price-feeds` and `decode-pnau-price-update` .

In `pyth-pnau-decoder-v2` , the `decode-and-verify-price-feeds` and `decode-pnau-price-update` functions can be merged, with the governance check:

```
  (try!
    (contract-call? .pyth-governance-v2 check-execution-flow contract-caller
  none))
```

inlined within a `let` variable. This removes one `begin` block and one inner call.

4. Standardize the codebase indentation level and switch to tabs:

- The `wormhole-core-v3: :parse-vaa` function is indented two levels more than other functions (after the first `let`).

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

Reduce the indentation by two levels.

- Refactor all operations that can be condensed into one line without affecting code readability:
  - In `wormhole-core-v3: :parse-and-verify-guardians-set`, the `ERR_GSU_CHECK_MODULE` and `ERR_GSU_CHECK_ACTION` `asserts` can each be written on one line.
  - In `pyth-pnau-decoder-v2:: check-merkle-proof`, the `result` can be written on a single line.

- Change all contract spaces from two spaces to one tab.

Recommendation

Implement the suggested changes.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Pyth Oracle 4
5. Risk Classification 5
5.1. Impact 5
5.2. Likelihood 5
5.3. Action required for severity levels 5
6. Security Assessment Summary 6
7. Executive Summary 7
8. Summary of Findings 8
8.1. High Findings 9
[H-01] Incorrect Processing of Price Batch Update Feeds 5 and 6 9
8.2. Low Findings 10
[L-01] Price Feed Update Verification Checks Incorrectly Skipped in Certain Batches 10
8.3. QA Findings 11
[QA-01] Redundant Return Data When Parsing PTGM 11
[QA-02] Reuse Offset Variable in Governance Data Source Parsing 12
[QA-03] Remove Debug Remnants 13
[QA-04] Redundant Return Data When Parsing Merkle Root Data 14
[QA-05] Redundant Merkle Ring Size Read Operation 15
[QA-06] Redundant Return Data When Parsing PNAU Header 16
[QA-07] Optimization of Parsing and Verifying Price Updates 17
[QA-08] Wormhole VAA Parsing Can Be Slightly Improved 20
[QA-09] Redundant Reading Operations When Parsing VAAs 21
[QA-10] Reuse Already Declared Local Variables When Recovering Public Key 22
[QA-11] Some Buffer Manipulation Function Calls Can Be Inlined 23
[QA-12] Improve Codebase Comments 25
[QA-13] Miscellaneous Codebase Improvements for Reducing Runtime Costs 26
[QA-14] Increment Contract Versions 28

[QA-14] Increment Contract Versions

Description

With the recent updates to the contracts, it is necessary to update all internal and contract name references to reflect the new versions.

Specifically:

- Change `wormhole-core-v3` to `wormhole-core-v4` , and update the internal comment from `Version: v3` to `Version: v4` .
- Change `pyth-governance-v2` to `pyth-governance-v3` , and update the internal comment from `Version: v2` to `Version: v3` .
- Change `pyth-pnau-decoder-v2` to `pyth-pnau-decoder-v3` , and update the internal comment from `Version: v2` to `Version: v3` .
- Change `wormhole-traits-v1` to `wormhole-traits-v2` , and update the internal comment accordingly.
- Change `pyth-oracle-v3` to `pyth-oracle-v4` , and update the internal comment accordingly.
- Change `pyth-storage-v3` to `pyth-storage-v4` , and update the internal comment accordingly.

Without these updates, the contracts cannot be deployed from the same principal as the previous versions, which could also lead to confusion for integrators.

Recommendation

Update all contract versions as necessary.