ClarityAlliance

HERMETICA USDH MINTING CONTRACT SECURITY REVIEW

Conducted by:
KRISTIAN APOSTOLOV, ALIN BARBATEI (ABA)

MARCH 20TH, 2025

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

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 Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

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 Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

3. Introduction

A time-boxed security review of Hermetica USDh, where Clarity Alliance reviewed the scope and provided insights on improving the protocol.

4. About Hermetica USDh

Hermetica's USDh is the first Bitcoin-backed synthetic dollar that yields up to 25%.

The Hermetica protocol couples spot BTC with a short perpetual futures position to create a synthetic dollar that is native to Bitcoin L1 and L2s.

Staked USDh, a Bitcoin backed, yield instruments accrues daily yields from perpetual futures funding rates.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

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 Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

6. Security Assessment Summary

Scope

The following contract was in the scope of the security review:

- `contracts/protocol/minting-auto-v1.clar`

Initial Commit Reviewed:
2d51015b223b844f9c3ed026669a97b1b594d41b

Final Commit After Audit Remediations:
863ab5f468ba5e76e4cc70721332f23d2884388e

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

7. Executive Summary

Over the course of the security review, Kristian Apostolov, Alin Barbatei (ABA) engaged with - to review Hermetica USDh. In this period of time a total of **13** issues were uncovered.

Protocol Summary

Protocol Name	Hermetica USDh
Date	March 20th, 2025

Findings Count

Severity	Amount
High	2
Medium	2
Low	1
QA	8
Total Findings	**13**

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

Summary of Findings

ID	Title	Severity	Status
[H-01]	Deprecated Pyth Oracle Version Is Used	High	Resolved
[H-02]	Hardcoded Pyth Price Exponent	High	Resolved
[M-01]	Pyth Price Confidence Is Not Validated	Medium	Resolved
[M-02]	Redeeming Incorrectly Consumes Minting Allowance	Medium	Resolved
[L-01]	Avoid Using tx-sender for Caller Identification	Low	Resolved
[QA-01]	Absence of Events for Critical Actions	QA	Resolved
[QA-02]	Typographical Error	QA	Resolved
[QA-03]	Redundant Tuple with a Single Element as Map Key	QA	Acknowledged
[QA-04]	Simplification of set-supported- asset	QA	Resolved
[QA-05]	Implement Standard Checks for All Saved Principals	QA	Resolved
[QA-06]	Unused Redeem Memo Argument	QA	Resolved
[QA-07]	Missing Required USDh Amount Validation	QA	Resolved
[QA-08]	Slippage Mechanism Is Ineffective Against Price Fluctuations	QA	Acknowledged

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
 [H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
 [M-01] Pyth Price Confidence Is Not Validated 11
 [M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
 [L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
 [QA-01] Absence of Events for Critical Actions 14
 [QA-02] Typographical Error 15
 [QA-03] Redundant Tuple with a Single Element as Map Key 16
 [QA-04] Simplification of set-supported- asset 17
 [QA-05] Implement Standard Checks for All Saved Principals 18
 [QA-06] Unused Redeem Memo Argument 19
 [QA-07] Missing Required USDh Amount Validation 20
 [QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

8. Findings

8.1. High Findings

[H-01] Deprecated Pyth Oracle Version Is Used

Description

In the `minting-auto-v1` contract, minting and redeeming can be performed using either the default 1:1 ratio or the Pyth Pull Oracle.

The Pyth contract currently in use, `'SP2T5JKWWP3FYYX4YRK8GK5BG2YCNGEAEY2P2PKN0.pyth-oracle-v2`, has been deprecated and archived since January 2025:

> This repo has been ARCHIVED. Please see https://github.com/Trust-Machines/stacks-pyth-bridge for the Stacks Pyth bridge.

Using this outdated, unaudited, and unmaintained contract may lead to the use of invalid price data.

Recommendation

Adopt the latest Pyth deployment contracts from the new `stacks-pyth-bridge` repository.

CONTENTS
1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[H-02] Hardcoded Pyth Price Exponent

Description

In the `minting-auto-v1` contract, minting and redeeming can be performed using either the default 1:1 ratio or the Pyth Pull Oracle.

The Pyth integration is defined as follows:

```
(decoded-price
  (match price-feed-bytes value
    (element-at (try!
      (contract-call? 'SP2T5JKWWP3FYYX4YRK8GK5BG2YCNGEAEY2P2PKN0.pyth-oracle-v2
       decode
    (
      some{conf:u0,
      ema-conf:u0,
      ema-price:0,
      expo:0,
      prev-publish-time:u0,
      price:
    ), price-identifier: 0x00, publish-time: (+ block-timestamp u1
  )
)
(price (to-uint (unwrap-panic (get price decoded-price))))
```

A significant issue exists in the current implementation of the protocol. The Pyth decoded price information returns an exponent (`expo`) and a `price` variable, but the actual price is determined by combining the two as indicated in their official documentation.

Since the minter contract directly uses the `price` element, it utilizes an incorrect price that does not account for decimals or the exponent. The current implementation also mistakenly assumes that the `oracle-base` (`10^8`) represents the price decimals.

Using an incorrect price will drastically alter the minted amount of tokens.

Recommendation

Convert the returned decoded price data to an `oracle-base` decimal value, if necessary, by checking against the `expo` variable.

An example of a Pyth price conversion function can be found in the Granite Protocol pyth-adapter.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

8.2. Medium Findings

[M-01] Pyth Price Confidence Is Not Validated

Description

In the `minting-auto-v1` contract, minting and redeeming can be performed using either the default 1:1 ratio or the Pyth Pull Oracle.

The Pyth integration is defined as follows:

```
(decoded-price
  (match price-feed-bytes value
    (element-at (try!
      (contract-call? 'SP2T5JKWWP3FYYX4YRK8GK5BG2YCNGEAEY2P2PKN0.pyth-oracle-v2
       decode
    (
      some{conf:u0,
      ema-conf:u0,
      ema-price:0,
      expo:0,
      prev-publish-time:u0,
      price:
    ), price-identifier: 0x00, publish-time: (+ block-timestamp u1
  )
)
(price (to-uint (unwrap-panic (get price decoded-price))))
```

Prices returned by the Pyth Network include a level of uncertainty, represented by a confidence interval.

Currently, the contract implementation does not validate the confidence level. It is essential to validate the confidence level to ensure that the price returned by the network falls within an acceptable range.

For example, a price for `STX` of `$3` with a confidence of `± $2` may be returned. In this scenario, the network is uncertain of the exact price, placing it within a `[$1, $5]` range.

Such a situation, while highly irregular, is still possible and could lead to financial loss for users if this price is used in collateral evaluation.

Recommendation

In the `minting-auto-v1`, implement a maximum confidence threshold (price/confidence) that is adjustable and checked when retrieving the price. Note that a confidence interval of 0 implies no spread in price and should be considered a valid price.

An example of a Pyth confidence interval check can be found in the Granite project's pyth-adapter.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[M-02] Redeeming Incorrectly Consumes Minting Allowance

Description

When `USDh` is minted using the `minting-auto-v1::mint` function, the minted amount is correctly subtracted from the current minting time window:

```
(ok (var-set current-mint-limit (-
  (get-current-mint-limit) amount-usdh-requested)))
```

However, when assets are redeemed by burning `USDh`, the mint limit is incorrectly reduced again.

This incorrect reduction of the limit during redemptions can prevent minters from minting additional tokens within the same window. Additionally, redeemers may face unnecessary delays, having to wait until the next time window to redeem their tokens.

Recommendation

Remove the line `(var-set current-mint-limit (- (get-current-mint-limit) amount-usdh-requested)))` from the `minting-auto-v1::redeem` function.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

8.3. Low Findings

[L-01] Avoid Using `tx-sender` for Caller Identification

Description

Within the contract, there are several instances where `tx-sender` is used instead of `contract-caller` or passing the caller's address.

This practice can lead to situations where minters or administrators, who fall victim to phishing scams, might unknowingly interact with malicious contracts. This could result in the execution of sensitive operations within the codebase.

For instance, if a minter interacts with a malicious contract, that contract could potentially mint tokens on their behalf using the `minting-auto-v1::mint` function.

It is important to note that such scenarios can only occur due to negligence on the part of the minter.

Recommendation

Replace all instances of `tx-sender` with `contract-caller` , except within the SIP-10 `transfer` function.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

8.4. QA Findings

[QA-01] Absence of Events for Critical Actions

Description

In the `minting-auto-v1` contract, when a significant variable is updated, no event is emitted to notify off-chain monitoring systems.

The absence of events complicates protocol tracking for any third-party systems.

Recommendation

Incorporate a `print` command to log both the previous and new values (where applicable) for all admin-restricted functions: `set-mint-limit` , `set-mint-limit-reset-window` , `set-block-delay` , `set-whitelist` , `set-custody-address` , and `set-supported-asset` .

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-02] Typographical Error

Description

In the `minting-auto-v1` contract, there is a typographical error in the comment preceding the `redeem` function. The word `Redemer` should be corrected to `Redeemer` .

Recommendation

Correct the typographical error as indicated.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
 [H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
 [M-01] Pyth Price Confidence Is Not Validated 11
 [M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
 [L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
 [QA-01] Absence of Events for Critical Actions 14
 [QA-02] Typographical Error 15
 [QA-03] Redundant Tuple with a Single Element as Map Key 16
 [QA-04] Simplification of set-supported- asset 17
 [QA-05] Implement Standard Checks for All Saved Principals 18
 [QA-06] Unused Redeem Memo Argument 19
 [QA-07] Missing Required USDh Amount Validation 20
 [QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-03] Redundant Tuple with a Single Element as Map Key

Description

In the `minting-auto-v1` contract, there is a map of supported assets:

```
(define-map supported-assets
  {
    contract: principal
  }
  {
    active: bool,
    price-feed-id: (buff 32),
    token-base: uint,
    slippage: uint,
  }
)
```

This map unnecessarily uses a tuple containing only one element, a `contract` principal, instead of using the principal directly. This approach increases overall operational costs and reduces code readability.

Recommendation

Modify the map to use a `principal` as the key instead of a tuple.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
 [H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
 [M-01] Pyth Price Confidence Is Not Validated 11
 [M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
 [L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
 [QA-01] Absence of Events for Critical Actions 14
 [QA-02] Typographical Error 15
 [QA-03] Redundant Tuple with a Single Element as Map Key 16
 [QA-04] Simplification of set-supported- asset 17
 [QA-05] Implement Standard Checks for All Saved Principals 18
 [QA-06] Unused Redeem Memo Argument 19
 [QA-07] Missing Required USDh Amount Validation 20
 [QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

ClarityAlliance
Security Review
Hermetica USDh

[QA-04] Simplification of set-supported-asset

Description

The `minting-auto-v1::set-supported-asset` function is used to configure information related to a supported token.

One of the settings passed is the `token-base`, which must always be equal to `10^asset_decimals`.

```
(asserts! (is-eq token-base (pow u10 (unwrap-panic
  (contract-call? token get-decimals)))) ERR_TOKEN_BASE_MISMATCH)
```

Since this condition is always necessary, passing the base itself is redundant. You can directly use `10^asset_decimals` as the base without needing to add any extra parameters to the function. .

Recommendation

Set the token base as `(pow u10 (unwrap-panic (contract-call? token get-decimals)))` in the `supported-assets` map.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-05] Implement Standard Checks for All Saved Principals

Description

Within the `minting-auto-v1` contract, sensitive principals are stored in the storage contracts. However, none of these principals are verified to ensure they conform to the standard of the current network.

If a testnet principal is mistakenly used instead of a mainnet principal, it could lead to critical functionality becoming inoperative.

Recommendation

Ensure that all storage contracts saving principals verify the validity of these principals for the current network by utilizing the `is-standard` function. .

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-06] Unused Redeem Memo Argument

Description

The `minting-auto-v1::redeem` function includes an optional `memo` argument that is never utilized. Although it is optional, providing this parameter does not influence the function's behavior.

Recommendation

Consider either removing the `memo` argument altogether or updating the `redeeming-reserve-v1::transfer` function to accept a memo and pass it along.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-07] Missing Required USDh Amount Validation

Description

In the `minting-auto-v1` contract, both the `mint` and `redeem` functions do not validate whether the `amount-asset-required` is greater than 0. If a value of 0 is passed, both operations will revert in the `usdh-token-v1` contract's internal token `mint` and `burn` functions with the error `(err 1)`.

This lack of validation complicates error debugging for external integrators.

Recommendation

In the `mint` and `redeem` functions of the `minting-auto-v1` contract, ensure that the `amount-asset-required` is checked to be greater than 0. If it is not, the operation should revert with a custom error code.

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations

Description

When a whitelisted user intends to mint or redeem, they provide a `slippage-tolerance` argument, which should indicate the maximum amount they are willing to lose due to fee differences or price fluctuations.
This `slippage-tolerance` argument is compared with the contract's stored `slippage` variable from the `supported-assets` map and is reverted if exceeded.

```
(slippage-bps (get slippage supported-asset-data))
;; ...
(asserts! (<= slippage-bps slippage-tolerance) ERR_SLIPPAGE_TOO_HIGH)
```

The issue with the current implementation is that the slippage is actually a percentage change in price variation, not an expected percentage change in the output amount:

```
(slippage-amount (/ (* price slippage-bps) bps-base))
(amount-asset-required (/ (* amount-usdh-requested oracle-base token-base)
  (- price slippage-amount) usdh-base))
```

Example scenario:

- Protocol and user slippage: 1000 (10%)
- `amount-usdh-requested` : 10,000 `USDh`
- Asset token: `aeUSDC`
- `amount-asset-required` is calculated as: `11,111`

```
= 10,000 / (1 - 0.1)
= 10,000 / 0.9
= 11,111
```

The requirement of 11,111 `aeUSDC` assets represents an 11.11% increase over the originally expected 10,000 amount (in the absence of a Pyth oracle), instead of a 10% slippage.

This issue is further exacerbated when the Pyth Oracle is used:

Example second scenario:

- Protocol and user slippage: 100 (1%)
- User wants to exchange `aeUSDC` to `USDh`
- User observes that the Pyth oracle shows a 1:1 exchange rate for the assets and initiates a request using an updated price feed
- User submits a mint request with `amount-usdh-requested` of 10,000 `USDh` , expecting to pay between [10,000 - 10,100] `aeUSDC` (his 1% slippage)
- Someone submits a more recent Pyth price update that spikes the exchange rate to 0.95 `aeUSDC` for 1 `USDh`

CONTENTS

1. About Clarity Alliance 2
2. Disclaimer 3
3. Introduction 4
4. About Hermetica USDh 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] Deprecated Pyth Oracle Version Is Used 9
[H-02] Hardcoded Pyth Price Exponent 10
8.2. Medium Findings 11
[M-01] Pyth Price Confidence Is Not Validated 11
[M-02] Redeeming Incorrectly Consumes Minting Allowance 12
8.3. Low Findings 13
[L-01] Avoid Using tx-sender for Caller Identification 13
8.4. QA Findings 14
[QA-01] Absence of Events for Critical Actions 14
[QA-02] Typographical Error 15
[QA-03] Redundant Tuple with a Single Element as Map Key 16
[QA-04] Simplification of set-supported- asset 17
[QA-05] Implement Standard Checks for All Saved Principals 18
[QA-06] Unused Redeem Memo Argument 19
[QA-07] Missing Required USDh Amount Validation 20
[QA-08] Slippage Mechanism Is Ineffective Against Price Fluctuations 21

- Since both the protocol and user slippage are 1%, they pass, regardless of how price fluctuation actually impacts the final amount
- `amount-asset-required` is calculated as: 10,632 `aeUSDC`

```
= 10,000 / (0.95 - 0.0095)
= 10,000 / 0.9405
= 10,632
```

Execution proceeds, and the user must pay 10,632 `aeUSDC` instead of their intended maximum of 10,100 `aeUSDC` to mint the 10,000 `USDh` , resulting in an unexpected increase of 532 `aeUSDC` .

Recommendation

Since the development team plans to configure the minter contract to accept Pyth prices no older than 2 blocks in production, along with an agreed-upon supported asset slippage with the main users, the issue can be acknowledged.

However, because the Pyth staleness check can be arbitrarily changed, leading to wider price variations, it is recommended to modify the contract, at least in future versions, as follows:

In the `mint` function, rename the `slippage-tolerance` variable to `maximum-assets-in` and ensure that the `amount-asset-required` is less than or equal to this value; otherwise, revert.

In the `redeem` function, rename the `slippage-tolerance` variable to `minimum-assets-out` and ensure that the `amount-asset-required` is greater than or equal to this value; otherwise, revert.