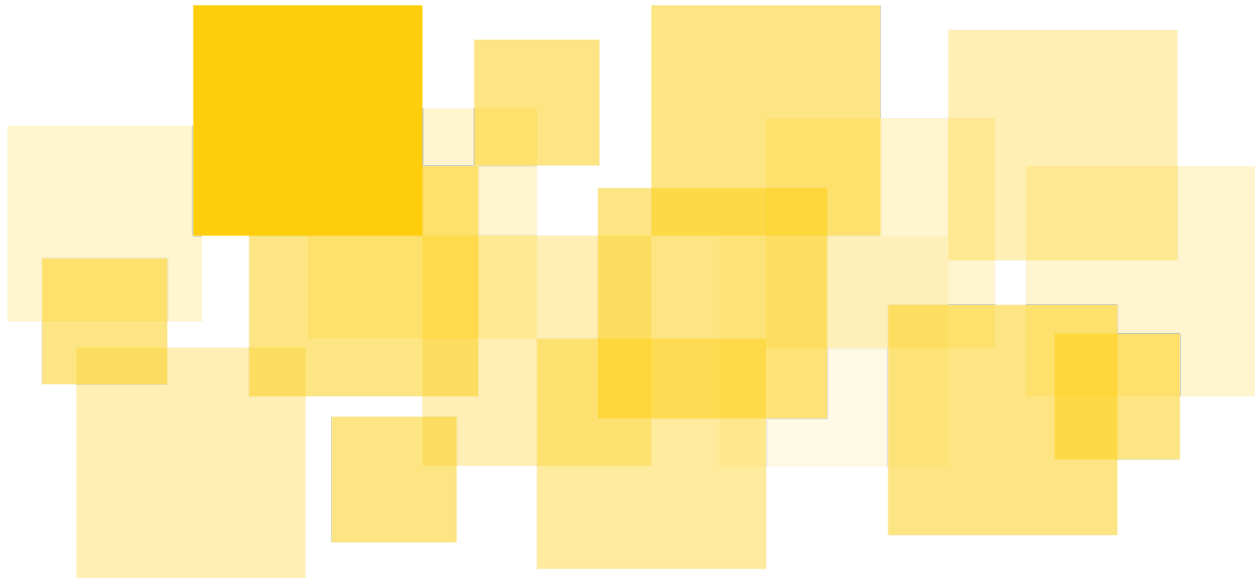


Security Audit Report

HydraDX - EMA Oracle Polkadot

Delivered: June 19, 2023



Prepared for HydraDX by





Table of Contents

- Executive Summary
- Goal
- Scope
- Platform Features Description
- Methodology
- Disclaimer
- Findings
 - [A1] Oracle is not updated by directly sending assets to an Omnipool
 - [A2] EMA of reciprocal price and reciprocal of EMA price diverges
- Informative Findings
 - [B1] `MAX_UNIQUE_ENTRIES` is defined but not used in the code
 - [B2] Variable naming does not reflect the purpose of the variables
 - [B3] Presence of identified potential vulnerabilities in dependencies of crates



Executive Summary

HydraDX engaged [Runtime Verification Inc.](#) to conduct a security audit of their pallets' code. The objective was to review the platform's business logic and implementations in Rust and identify any issues that could cause the system to malfunction or be exploited.

The audit was conducted over the course of 3 calendar weeks (May 3, 2023 through May 24, 2023) and focused on analyzing the security of the source code of HydraDX's EMA oracle system, which enables users to query information about asset exchange rates as well as monitor exchange rate sources for updated information.

The audit led to identifying issues of potential severity for the protocol's health, which have been identified as follows:

- Exchange rate manipulation: [A1](#)
- Functional correctness: [A2](#)

In addition, several informative findings and general recommendations also have been made, including:

- Potentially compromised dependencies: [B3](#)
- Best practices: [B2](#), [B3](#)
- Code optimization-related particularities: [B1](#)

All the potentially critical issues have been addressed, while a substantial amount of the informative findings and general recommendations have been incorporated into the platform's code. All of the remaining findings have been acknowledged by the client and deemed non-threatening to the integrity of the platform, when not intended by design.



Goal

The goal of the audit is threefold:

1. Review the high-level business logic (protocol design) of HydraDX's system based on the provided documentation;
2. Review the low-level implementation of the system for the individual Rust modules (Oracle and math modules);
3. Analyze the integration between the multiple modules in the scope of the engagement and reason about possible exploitive corner cases.

The audit focuses on identifying issues in the system's logic and implementation that could potentially render the system vulnerable to attacks or cause it to malfunction. Furthermore, the audit highlights informative findings that could be used to improve the safety and efficiency of the implementation.



Scope

The scope of the audit is limited to the code contained in repositories of two functional modules provided by the client. These projects are:

1. EMA Oracle Pallet ([Commit dd886469373b9133cbf0ae705ac354e93db2c7b7](#)):
implements the functionalities related monitoring of operations influencing the exchange rate of assets, as well as computing and providing the calculated asset price to applications;
2. EMA Math Module ([Commit 6bd2f8fb42f83735d8cc3fe16c488b8ec134196a](#)):
implements the necessary operations for calculating the asset price according to operations being performed in the exchange rate source and the progression of time;

The comments provided in the code, and a general description of the project, including samples of tests used for interacting with the platform, and online documentation provided by the client were used as reference material.

The audit is limited in scope to the artifacts listed above. Off-chain, auto-generated, or client-side portions of the codebase, as well as deployment and upgrade scripts, are not in the scope of this engagement.

Commits addressing the findings presented in this report have also been analyzed to ensure the resolution of potential issues in the protocol.



Platform Features Description

Aiming to improve oracle services to the Polkadot ecosystem, the HydraDX EMA oracle implements an L1 oracle chain focused on readily providing pricing information using the latest trades of oracle sources.

Monitoring the blockchain for events that cause price modification of assets, the EMA oracle groups these events into accumulators, which are used for aiding the definition of the exchange for an asset within the progression of a block. This strategy allows for reliably obtaining prices of assets without suffering from the influence of individual, minor operations.

The ema oracle pallet works by interpolating the spot price at the end of each block and calculating the ema price. Specifically, in a block, other pallets will feed the ema with the spot prices by calling `on_trade` or `on_liquidity_changed` hooks. For each token pair, the accumulator in the ema oracle will only record the latest price. At the end of each block, the `on_finalize` function will be invoked to calculate the new ema price for each token pair. The new ema price is defined as:

$$\begin{aligned} \text{ema}_{\text{new}}(\text{tokenA}, \text{tokenB}, \text{period}) &= (1 - \alpha(\text{period})) * \\ &\text{ema}_{\text{old}}(\text{tokenA}, \text{tokenB}) + \alpha(\text{period}) * \text{spotPrice}(\text{tokenA}, \text{tokenB}) \\ \alpha(\text{period}) &= 2 / (\text{period} + 1) \end{aligned}$$

The spot price of the token pair is either the latest spot price recorded in the accumulator or the spot price from the last block if there is no trade for the token pair in the current block.

All the necessary software and information to utilize this oracle mechanism is provided by the HydraDX team.



Methodology

Although the manual code review cannot guarantee to find all possible security vulnerabilities as mentioned in our [Disclaimer](#), we have followed the approaches described below to make our audit as thorough as possible.

First, we rigorously reasoned about the business logic of the code, validating security-critical properties to ensure the absence of loopholes in the business logic. To this end, we carefully analyzed all the proposed features of the platform and actors involved in the lifetime of a deployed contract.

Second, we thoroughly reviewed the pallet source code to detect any unexpected (and possibly exploitable) behaviors. To facilitate our understanding of the platform's behavior, higher-level representations of the Rust codebase were created, where the most comprehensive were:

- Modeled sequences of mathematical operations as equations and, considering the limitations of size and types of variables of the utilized modules, checking if all desired properties hold for any possible input value.

This approach enabled us to systematically check consistency between the logic and the provided Rust implementation of the pallet.

Tools capable of identifying dependency-related issues such as cargo-audit and cargo-geiger have been used to analyze possible security issues in crates referenced in this code.

Finally, we performed rounds of internal discussion with security experts over the code and platform design, aiming to verify possible exploitation vectors and to identify improvements for the analyzed contracts.

Additionally, given the nascent Polkadot development and auditing community, we reviewed [this list](#) of known Ethereum security vulnerabilities and attack vectors and checked whether they apply to the smart contracts and scripts; if they apply, we checked whether the code is vulnerable to them.



Disclaimer

This report does not constitute legal or investment advice. You understand and agree that this report relates to new and emerging technologies and that there are significant risks inherent in using such technologies that cannot be completely protected against. While this report has been prepared based on data and information that has been provided by you or is otherwise publicly available, there are likely additional unknown risks which otherwise exist. This report is also not comprehensive in scope, excluding a number of components critical to the correct operation of this system. This report is for informational purposes only and is provided on an "as-is" basis and you acknowledge and agree that you are making use of this report and the information contained herein at your own risk. The preparers of this report make no representations or warranties of any kind, either express or implied, regarding the information in or the use of this report and shall not be liable to you or any third parties for any acts or omissions undertaken by you or any third parties based on the information contained herein.

Smart contracts are still a nascent software arena, and their deployment and public offering carries substantial risk.

Finally, the possibility of human error in the manual review process is very real, and we recommend seeking multiple independent opinions on any claims which impact a large quantity of funds.



Findings

Findings presented in this section are issues that can cause the system to fail, malfunction, and/or be exploited, and should be properly addressed.

All findings have a severity level and an execution difficulty level, ranging from low to high, as well as categories in which it fits.

[A1] Oracle is not updated by directly sending assets to an Omnipool

Severity: Medium

Difficulty: Low

Recommended Action: Fix Design

Addressed By Client

Description

Idealized to have HydraDX's Omnipools as the source of the exchange rates, the EMA oracle works by setting hooks that watch operations being performed in the liquidity pools and, according to the exchange, deposit, and withdrawal operations, modify the respective assets' exchange rate. In other words, the EMA oracle is lazily updated, creating the necessity for certain events to be triggered in order to perform a price update.

While comprehensive, the set of events that trigger price updates in the oracle does not include direct transfers of assets to Omnipools, as no events are triggered in such a scenario. This means that direct transfers to the liquidity pools can influence the price of assets without the oracle being aware of it, thus creating a situation where the exchange rate provided can be considered misleading.

Furthermore, malicious users could attempt to modify the price of assets in Omnipools using direct transfers in order to influence the capability of arbitrageurs and other honest users of performing operations in moments of high congestion of the network, and, for instance, prevent rightful liquidations from happening in lending protocols.

Recommendation

In order to correctly provide the exchange rate of assets, one of the approaches provided below must be followed:

- The design of the oracle must be changed, making it not lazily updated, or;
- The oracle must be aware of the direct transfer of assets from users to Omnipools, or;
- The Omnipools should not be able to receive direct transfers.

Status

The client has addressed this finding by disabling the possibility of users sending assets to Omnipools via direct transfers, outside of the scope of this engagement.

[A2] EMA of reciprocal price and reciprocal of EMA price diverges

Severity: Medium

Difficulty: Medium

Recommended Action: Fix Code

Not Addressed By Client

Description

Considering the price p as the outcome of the following equation: $\frac{reserve_A}{reserve_B}$. The implementation interpolates the price p at the end of each block and, subsequently, calculates the ema price $ema(p)$. When calculating the EMA of the reciprocal price, the implementation directly returns $\frac{1}{ema(p)}$ instead of interpolating $\frac{1}{p}$ at the end of each block, which is followed by the calculation of the ema of the reciprocal price $ema(\frac{1}{p})$.

We know that $\frac{1}{ema(p)} \neq ema\left(\frac{1}{p}\right)$, which indicates a incorrect calculation of the reciprocal EMA.

This may lead end users to be provided with erroneous values.

Recommendation

The implementation should Interpolate the reciprocal price $\frac{1}{p}$ at the end of each block and calculate the ema price separately.

Status

The client has acknowledged this finding.



Informative Findings

The findings presented in this section do not necessarily represent any flaw in the code itself. However, they indicate areas where the code may need external support or deviates from best practices. We have also included information on potential code size reductions and remarks on the operational perspective of the contract.



[B1] `MAX_UNIQUE_ENTRIES` is defined but not used in the code

Severity: Informative

Recommended Action: Fix Code

Not Addressed By Client

Description

The EMA oracle groups the data of triggered events (trades, supply, and withdrawal operations) in structures referred to as Accumulators. These structures are used to simplify the data storage and calculation of the exchange rate of asset pairs, where one Accumulator is created to handle the operations related to each individual token pair that had events triggered in a single block.

To prevent issues of resource usage/blockchain congestion, the EMA oracle establishes the maximum number of accumulators that can exist simultaneously in a single block. As documented in the code, the constant `MAX_UNIQUE_ENTRIES` is defined, and supposed to serve this purpose, but it ends up not being used in the code.

Recommendation

Remove the `MAX_UNIQUE_ENTRIES` definition, or enforce `MaxUniqueEntries` to have the value configured in this constant.

Status

The client acknowledged this finding.



[B2] Variable naming does not reflect the purpose of the variables

Severity: Informative

Recommended Action: Fix Code

Not Addressed By Client

Description

In different instances of the code for HydraDX's EMA oracle, some of the variable names do not necessarily reflect the purpose used by them. One example is the `timestamp` variable, of the `OracleEntry` struct, which, while used to represent a moment in time that the oracle entry represents, the type of this variable as well as the value that it receives is representative of a block number. The `timestamp` variable could be renamed to, for instance, `entry_block_number` or simply `block_number`.

Another example of this is seen in the variable `parent` (line 311, `warehouse/ema-oracle/src/lib.rs`), which is used to represent the block number prior to the one currently being built. A more meaningful variable name could be, for instance, `previous_block_number` or `last_completed_block_number`.

Recommendation

As a best-practice recommendation, variables should have names that prevent misunderstandings about their meaning, and that are reflective of their purpose in the code.

Status

The client has partially addressed this finding.

[B3] Presence of identified potential vulnerabilities in dependencies of crates

Severity: Informative

Recommended Action: Fix Code

Not Addressed By Client

Description

By using the cargo-audit functionality, it is possible to match the analyzed code against the RUSTSEC security advisory database to identify possible exploitation vectors or code improvements related to dependencies of the code. Using this functionality in the audited repositories, the following lists of potential vulnerabilities and warnings, alongside reference to more information, were found:

- Related to the math pallet, under the HydraDX-math repository:

```
Crate:    remove_dir_all
Version:  0.5.3
Title:    Race Condition Enabling Link Following and Time-of-check Time-of-
use (TOCTOU)
Date:     2023-02-24
ID:       RUSTSEC-2023-0018
URL:      https://rustsec.org/advisories/RUSTSEC-2023-0018
Solution: Upgrade to >=0.8.0
```

```
Crate:    atty
Version:  0.2.14
Warning:  unsound
Title:    Potential unaligned read
Date:     2021-07-04
ID:       RUSTSEC-2021-0145
URL:      https://rustsec.org/advisories/RUSTSEC-2021-0145
```

```
Crate:    crossbeam-channel
Version:  0.5.6
Warning:  yanked (author declared that the package should not be used)
```

- Related to the EMA oracle pallet, under the warehouse repository:

Crate: remove_dir_all
Version: 0.5.3
Title: Race Condition Enabling Link Following and Time-of-check Time-of-use (TOCTOU)
Date: 2023-02-24
ID: RUSTSEC-2023-0018
URL: <https://rustsec.org/advisories/RUSTSEC-2023-0018>
Solution: Upgrade to >=0.8.0


Crate: time
Version: 0.1.45
Title: Potential segfault in the time crate
Date: 2020-11-18
ID: RUSTSEC-2020-0071
URL: <https://rustsec.org/advisories/RUSTSEC-2020-0071>
Severity: 6.2 (medium)
Solution: Upgrade to >=0.2.23

Crate: wasmtime
Version: 0.38.3
Title: Bug in Wasmtime implementation of pooling instance allocator
Date: 2022-11-10
ID: RUSTSEC-2022-0076
URL: <https://rustsec.org/advisories/RUSTSEC-2022-0076>
Severity: 7.4 (high)
Solution: Upgrade to >=1.0.2, <2.0.0 OR >=2.0.2

Crate: ansi_term
Version: 0.12.1
Warning: unmaintained
Title: ansi_term is Unmaintained
Date: 2021-08-18
ID: RUSTSEC-2021-0139
URL: <https://rustsec.org/advisories/RUSTSEC-2021-0139>

Crate: mach
Version: 0.3.2
Warning: unmaintained
Title: mach is unmaintained
Date: 2020-07-14
ID: RUSTSEC-2020-0168
URL: <https://rustsec.org/advisories/RUSTSEC-2020-0168>

Crate: parity-util-mem
Version: 0.11.0
Warning: unmaintained



```
Title:    parity-util-mem Unmaintained
Date:    2022-11-30
ID:      RUSTSEC-2022-0080
URL:     https://rustsec.org/advisories/RUSTSEC-2022-0080
```

```
Crate:    parity-wasm
Version:  0.42.2
Warning:  unmaintained
Title:    Crate `parity-wasm` deprecated by the author
Date:     2022-10-01
ID:       RUSTSEC-2022-0061
URL:      https://rustsec.org/advisories/RUSTSEC-2022-0061
```

```
Crate:    atty
Version:  0.2.14
Warning:  unsound
Title:    Potential unaligned read
Date:     2021-07-04
ID:       RUSTSEC-2021-0145
URL:      https://rustsec.org/advisories/RUSTSEC-2021-0145
```

```
Crate:    crossbeam-channel
Version:  0.5.6
Warning:  yanked (author declared that the package should not be used)
```

All the referred links were last accessed on May 31st, 2023.

Recommendation

Ideally, the crates which are triggering such warnings could be either updated or replaced with adequate ones, depending on the scenario, risk, and available solution, but it is important to highlight that **the dependencies that are triggering such issues and warnings are from Polkadot's substrate itself**. In other words, **the solutions to the issues mentioned above are outside the HydraDX team's current scope, as they are part of the infrastructure required for constructing such pallets**.

Status

The finding has been acknowledged by the client.