

ABRACADABRA MONEY · SECURITY REVIEW

Abracadabra Money MIMSwap Updates.

A security review of MIMSwap Updates.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
Abracadabra Money	MIMSwap Updates	Guardian	January 20 – January 24, 2025	January 24, 2025

§ 00

Contents.

23 pages: overview, confidence ranking, scope, 12 findings, disclaimer, and about Guardian.

§ 01	Overview	P. 03
§ 02	Confidence Ranking	P. 04
§ 03	Scope & Classifications	P. 05
	Files in Scope	P. 06
§ 04	Findings	P. 07
	Main Review · 12 findings	P. 08
§ 05	Disclaimer	P. 22
§ 06	About Guardian	P. 23

§ 01

Overview.

Guardian reviewed MIMSwap Updates across a single review round. The final report includes 5 Critical, 1 High, 3 Medium, 3 Low findings.

AUDIT FIRM	REPOSITORY
Guardian	https://github.com/Abracadabra-money/abracadabra-money-contracts
CLIENT	METHODOLOGY
Abracadabra Money	Smart contract · Manual · Main Review
PROJECT	MAIN REVIEW COMMIT
MIMSwap Updates	968993d97f31
PREPARED BY	
Guardian	
AUDIT WINDOW	
January 20 – January 24, 2025	
FINAL REPORT	
January 24, 2025	

1.1 Confidence Ranking

3 / 5

Moderate Confidence

Guardian assigns a Confidence Ranking of 3 out of 5, combining finding severity, remediation outcome, and the reviewer’s assessment of latent risk. The full scale is on the next page.

1	2	3	4	5
VERY LOW CONFIDENCE	LOW CONFIDENCE	MODERATE CONFIDENCE	HIGH CONFIDENCE	VERY HIGH CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED
■ Critical	5	5	0	0	0
■ High	1	1	0	0	0
■ Medium	3	3	0	0	0
■ Low	3	3	0	0	0
■ Informational	0	0	0	0	0
All severities	12	12	0	0	0

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence	Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity findings.
3 Moderate Confidence ASSIGNED	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed MIMSwap Updates in <https://github.com/Abracadabra-money/abracadabra-money-contracts>. The scope includes 555 source lines across 2 files.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH

Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.

IMPACT · MEDIUM

Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.

IMPACT · LOW

Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.

LIKELIHOOD · HIGH

Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.

LIKELIHOOD · MEDIUM

Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.

LIKELIHOOD · LOW

Unlikely to ever occur in production.

Files in Scope.

FILE	NSLOC	TOTAL
src/mimswap/MagicLP.sol	500	671
src/oracles/aggregators/MagicLpAggregator.sol	55	84

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
C-01	LP Prices Can Be Easily Manipulated After The First buyShares Call	Logical Error	CRITICAL	PENDING	p. 8
C-02	Flawed ABOVE_ONE Case	Logical Error	CRITICAL	PENDING	p. 10
C-03	LP Prices Can Be Easily Manipulated After The First buyShares Call	Logical Error	CRITICAL	PENDING	p. 11
C-04	Stable Pool Pricing Used For Volatile Assets	Logical Error	CRITICAL	PENDING	p. 13
C-05	Incorrect Pricing Formulas Used	Logical Error	CRITICAL	PENDING	p. 14
H-01	k=0 Case Not Handled	Logical Error	HIGH	PENDING	p. 15
M-01	Lack of Staleness Checks for Oracle Data	Validation	MEDIUM	PENDING	p. 16
M-02	Lack of Staleness Checks for Oracle Data	Validation	MEDIUM	PENDING	p. 17
M-03	Risk Of Overflow On Sqrt Calculation	Overflow	MEDIUM	PENDING	p. 18
L-01	Inaccurate Event Emission	Events	LOW	PENDING	p. 19
L-02	setParameters Stepwise Jump in Lp Price	Validation	LOW	PENDING	p. 20
L-03	correctRState LP Price Manipulation	Warning	LOW	PENDING	p. 21

CRITICAL C-01

PENDING

LP Prices Can Be Easily Manipulated After The First buyShares Call

CATEGORY	Logical Error
LOCATION	MagicLp.sol, MagicLpAggregator.sol
REVIEW	Main Review

DESCRIPTION

In the DODO-like PMM logic of `MagicLP`, the contract maintains a state variable `_RState_` (`ONE`, `ABOVE_ONE`, or `BELOW_ONE`) to reflect whether the pool's actual ratio of base vs. quote is near, above, or below the "target ratio" `_I_`.

However, when seeding liquidity via `buyShares`, the function does not recalculate `_RState_` using the PMM formulas. Instead, if it's the initial deposit (i.e., `totalSupply == 0`), it leaves `_RState_` set to `ONE` by default. And for subsequent calls, it leaves `_RState_` unchanged. This leads to:

- A mismatch if the deposited ratio is nowhere near the target ratio `_I_`. The pool's actual ratio may be well above or below `_I_`, yet `_RState_` remains set to `ONE`.
- Incorrect `MagicLpAggregator` calculations: The aggregator sees `BASE_TARGET/QUOTE_TARGET` that were set under the assumption `_RState_ = ONE` and `buyShares` ratio matched `_I_`. As a result, it returns a wrong LP price until a trade (e.g., `sellBase` or `sellQuote`) forces the PMM logic to recalculate `_RState_` properly.

Example scenario:

- The pool is intended for a 1:1 ratio (`_I_() = 1`) but the user deposits 4.537 base tokens and 12.524 quote tokens, an actual ratio $\sim 1:2.76$.
- `buyShares` sees `totalSupply == 0` (first deposit) and sets `_RState_` set to `ONE`, ignoring the fact that the real ratio is $\sim 2.76:1$.
- The aggregator (which references `_BASE_TARGET_`, `_QUOTE_TARGET_` from a "ONE" perspective) might report an LP price ~ 2.0 , whereas a naive sum-of-reserves would yield ~ 3.7 .
- Once a small trade (`sellBase`) occurs, the PMM code calculates the real ratio and transitions `_RState_` to `BELOW_ONE` or `ABOVE_ONE`, causing a big jump in the aggregator's reported LP price.

RECOMMENDATION

Consider recalculating `** _RState_ (**and** _BASE_TARGET_ **, _QUOTE_TARGET_ **)` in the `buyShares` and `sellShares` functions.

C-01 LP Prices Can Be Easily Manipulated After The First buyShares Call

CONTINUED

— RESOLUTION

Status: Pending.

CRITICAL C-02

PENDING

Flawed ABOVE_ONE Case

CATEGORY	Logical Error
LOCATION	MagicLpAggregator.sol
REVIEW	Main Review

DESCRIPTION

The current pricing mechanism can lead to an R_STATE being calculated that does not provide an accurate representation for the R_STATE of the pool and leads to unexpected pricing results and reverts.

Consider the following:

- (1) RState.ABOVE_ONE indicates a shortage of base relative to base target, and a surplus of quote relative to quote target.
 - (2) $((\text{quoteTargetNormalized} * r) / i) - ((Q * r) / i)$ within the latestAnswer function is bound to be negative since $Q > \text{quoteTargetNormalized}$ when RState.ABOVE_ONE
 - (3) With a small baseTargetNormalized, the calculation for B will ultimately underflow and DoS the LP pricing mechanism entirely.
- By mixing and match the current pool targets and mapping them to reserves with external prices, this revert is bound to happen.

RECOMMENDATION

Ensure that in the ABOVE_ONE case the baseTargetNormalized exceeds the difference between $((\text{quoteTargetNormalized} * r) / i) - ((Q * r) / i)$.

PROOF OF CONCEPT <https://gist.github.com/DanielGelfand/333dc14b345dae088d3afb91cf87cef5>

RESOLUTION

Status: Pending.

CRITICAL C-03

PENDING

LP Prices Can Be Easily Manipulated After The First buyShares Call

CATEGORY	Logical Error
LOCATION	MagicLp.sol, MagicLpAggregator.sol
REVIEW	Main Review

DESCRIPTION

In the DODO-like PMM logic of `MagicLP`, the contract maintains a state variable `_RState_` (`ONE`, `ABOVE_ONE`, or `BELOW_ONE`) to reflect whether the pool's actual ratio of base vs. quote is near, above, or below the "target ratio" `_I_`.

However, when seeding liquidity via `buyShares`, the function does not recalculate `_RState_` using the PMM formulas. Instead, if it's the initial deposit (i.e., `totalSupply == 0`), it leaves `_RState_` set to `ONE` by default. And for subsequent calls, it leaves `_RState_` unchanged. This leads to:

- A mismatch if the deposited ratio is nowhere near the target ratio `_I_`. The pool's actual ratio may be well above or below `_I_`, yet `_RState_` remains set to `ONE`.
- Incorrect `MagicLpAggregator` calculations: The aggregator sees `BASE_TARGET/QUOTE_TARGET` that were set under the assumption `_RState_ = ONE` and `buyShares` ratio matched `_I_`. As a result, it returns a wrong LP price until a trade (e.g., `sellBase` or `sellQuote`) forces the PMM logic to recalculate `_RState_` properly.

Example scenario:

- The pool is intended for a 1:1 ratio (`_I_() = 1`) but the user deposits 4.537 base tokens and 12.524 quote tokens, an actual ratio $\sim 1:2.76$.
- `buyShares` sees `totalSupply == 0` (first deposit) and sets `_RState_` set to `ONE`, ignoring the fact that the real ratio is $\sim 2.76:1$.
- The aggregator (which references `_BASE_TARGET_`, `_QUOTE_TARGET_` from a "ONE" perspective) might report an LP price ~ 2.0 , whereas a naive sum-of-reserves would yield ~ 3.7 .
- Once a small trade (`sellBase`) occurs, the PMM code calculates the real ratio and transitions `_RState_` to `BELOW_ONE` or `ABOVE_ONE`, causing a big jump in the aggregator's reported LP price.

RECOMMENDATION

Consider recalculating `** _RState_ (**and** _BASE_TARGET_ **, _QUOTE_TARGET_ **)` in the `buyShares` and `sellShares` functions.

C-03 LP Prices Can Be Easily Manipulated After The First buyShares Call**CONTINUED**

PROOF OF CONCEPT <https://gist.github.com/DanielGelfand/7ac7deeb8bc0eb3212e2e792cf618118>

— RESOLUTION

Status: Pending.

CRITICAL C-04

PENDING

Stable Pool Pricing Used For Volatile Assets

CATEGORY	Logical Error
LOCATION	MagicLp.sol
REVIEW	Main Review

DESCRIPTION

This update was meant to support volatile assets in the LP pair, but the current pool does not use an external oracle to dictate the equilibrium price `_I_`.

Without protocol owner pausing and intervening, over time the `_I_` will be become stale, and lead to inaccurate regression targets and pricing within the pool for users.

For example, consider the case where token X is initially priced at `\$1`, the `_I_` guiding price is set to `\$1`, but then the token performs a 2x and becomes `\$2`. The price will be lower than market price, as the `_I_` value is stale ($P = iR$), and tokens can be extracted at a cheaper cost.

RECOMMENDATION

Consider enabling the usage of an external oracle to dictate the `_I_` value when calculating the quadratic solutions, as that is the most up-to-date value for the asset.

RESOLUTION

Status: Pending.

CRITICAL **C-05**

PENDING

Incorrect Pricing Formulas Used

CATEGORY	Logical Error
LOCATION	MagicLPAggregator.sol
REVIEW	Main Review

DESCRIPTION

In the `RState.BELOW_ONE` case within the `MagicLPAggregator`, the reserve formula for the `RState.ABOVE_ONE` case is used.

Vice versa, for the `RState.ABOVE_ONE` case within the `MagicLPAggregator`, the reserve formula for the `RState.BELOW_ONE` case is used.

This will lead to inaccurate reserve calculations and inaccurate collateral pricing in the Cauldron.

RECOMMENDATION

Use the correct formulas. See [<https://docs.dodoex.io/en/product/pmm-algorithm/details-about-pmm>] (<https://docs.dodoex.io/en/product/pmm-algorithm/details-about-pmm>) for the formulas necessary.

The simplest solution is to just edit the inequality to use the right cases:

```
quoteTargetNormalized.divWad(baseTargetNormalized) >  
baseAnswerNormalized.divWad(quoteAnswerNormalized)
```

RESOLUTION

Status: Pending.

HIGH **H-01**

PENDING

k=0 Case Not Handled

CATEGORY	Logical Error
LOCATION	MagicLpAggregator.sol
REVIEW	Main Review

DESCRIPTION

When `k = 0` within the MagicLP, the price of the pair stays fixed at the initial guide price `i`, which is a valid pricing curvature for the pair.

Within the MagicLpAggregator, the solution for `B` in the BELOW_ONE case would use `ikpq.divWad(ikpq + baseAnswerNormalized - ipq)`, but if `k = 0` then it would ultimately turn into `0 / (0 + baseAnswerNormalized - ipq)`

If `baseAnswerNormalized == ipq`, then there will be a division by zero, the aggregator will revert, and consequently users may not be able to lend in the Cauldron or receive their collateral depending on when `k` was modified.

RECOMMENDATION

Consider enforcing that `k != 0` within the MagicLP to entirely avoid this scenario.

RESOLUTION

Status: Pending.

MEDIUM **M-01**

PENDING

Lack of Staleness Checks for Oracle Data

CATEGORY Validation
LOCATION MagicLpAggregator.sol
REVIEW Main Review

DESCRIPTION

The `MagicLpAggregator` contract calls `latestAnswer` directly for both `baseOracle` and `quoteOracle` without verifying whether these oracles have recent and valid price data. Moreover, according to Chainlink's documentation, the `latestAnswer` function is deprecated. `latestAnswer` doesn't throw an error when there is no answer but returns `0`.

RECOMMENDATION

It is recommended to ensure the prices retrieved from the price feeds are not too old (e.g., older than X seconds). Also, it is recommended to enforce that the price is higher than 0 to ensure that it is a valid price. To get proper access to all these data, the `latestRoundData` function should be used instead of the deprecated `latestAnswer`.

Example code snippet:

```
``solidity
(, int256 price, , uint256 updatedAt, ) = baseOracle.latestRoundData();
require(block.timestamp - updatedAt <= MAX_STALE_PERIOD, "Stale oracle price");
require(price > 0, "Invalid price");
``
```

RESOLUTION

Status: Pending.

MEDIUM **M-02**

PENDING

Lack of Staleness Checks for Oracle Data

CATEGORY	Validation
LOCATION	MagicLpAggregator.sol
REVIEW	Main Review

DESCRIPTION

The `MagicLpAggregator` contract calls `latestAnswer` directly for both `baseOracle` and `quoteOracle` without verifying whether these oracles have recent and valid price data. Moreover, according to Chainlink's documentation, the `latestAnswer` function is deprecated. `latestAnswer` doesn't throw an error when there is no answer but returns `0`.

RECOMMENDATION

It is recommended to ensure the prices retrieved from the price feeds are not too old (e.g., older than X seconds). Also, it is recommended to enforce that the price is higher than 0 to ensure that it is a valid price. To get proper access to all these data, the `latestRoundData` function should be used instead of the deprecated `latestAnswer`.

Example code snippet:

```
``solidity
(, int256 price, , uint256 updatedAt, ) = baseOracle.latestRoundData();
require(block.timestamp - updatedAt <= MAX_STALE_PERIOD, "Stale oracle price");
require(price > 0, "Invalid price");
``
```

RESOLUTION

Status: Pending.

MEDIUM **M-03**

PENDING

Risk Of Overflow On Sqrt Calculation

CATEGORY Overflow
LOCATION Math.sol: 79
REVIEW Main Review

DESCRIPTION

Within the usage of more volatile assets, there is now a greater risk of overflow when solving the quadratic functions.

For example, function `_SolveQuadraticFunctionForTarget` risks overflow within the else-if case: `else if ((ki * delta) / ki == delta)`

Depending on the K and I values (which can be up to 10^{36} and 10^{18}) respectively, the multiplication with a large delta will overflow, which effects the `adjustedTarget` function and hence most protocol actions.

Another example is `uint256 squareRoot = DecimalMath.mulFloor((DecimalMath.ONE - k) * 4, DecimalMath.mulFloor(k, V0) * V0);` within `_SolveQuadraticFunctionForTrade` which can overflow when multiplying `K * V0 * V0` which ultimately leads to failed trades.

RECOMMENDATION

Consider using libraries that support intermediate overflows. Furthermore, clearly document the risks with using large K values, and tokens with large decimals.

RESOLUTION

Status: Pending.

LOW **L-01**

PENDING

Inaccurate Event Emission

CATEGORY Events
LOCATION MagicLp.sol: 552
REVIEW Main Review

DESCRIPTION

Function `setTargets` emits `TargetChanged(baseTarget, quoteTarget)` but the pool's new base and quote targets can be different than the parameters passed in due to the `_adjustTarget()` call, which may shift the targets depending on whether the pool is long base or quote.

RECOMMENDATION

Clearly document this behavior or adjust the event to use the actual `_BASE_TARGET_` and `_QUOTE_TARGET_` post-adjustment.

RESOLUTION

Status: Pending.

LOW **L-02**

PENDING

setParameters Stepwise Jump in Lp Price

CATEGORY Validation
LOCATION MagicLp.sol
REVIEW Main Review

— DESCRIPTION

When the `MagicLP` owner or an authorized operator, uses the `setParameters` function to update the pool parameters and then almost immediately unpauses or reactivates the pool, there will be an abrupt jump in the aggregator's reported LP price as the aggregator's math depends on those key parameters. The `setParameters` call is expected to be executed frequently in protocol owned pools. This call or, which is the same, this jump in the aggregator's reported LP price, could be easily sandwiched for a profit.

— RECOMMENDATION

Be aware of this constrain whenever the `MagicLpAggregator` is integrated and used by any other protocol.

— RESOLUTION

Status: Pending.

LOW **L-03**

PENDING

correctRState LP Price Manipulation

CATEGORY	Warning
LOCATION	MagicLp.sol
REVIEW	Main Review

— DESCRIPTION

When a user takes a flashloan, it opens a precarious situation for the protocol since the reserves leave the LP temporarily, and if the targets were to be adjusted this would manipulate the price of the LP collateral. An attack path such as `flashloan -> sync` is prevented because function `sync` has the `nonReentrant` modifier, but `correctRState` does not. Although `correctRState` is usually not triggered as it requires a rare protocol state, it is best to prevent reentrancy targets entirely given how adjustments in the targets will influence the valuation of collateral within the Cauldron.

— RECOMMENDATION

Add a `nonReentrant` modifier to function `correctRState`. Furthermore, document that calling `correctRState` directly will affect the valuation of collateral.

— RESOLUTION

Status: Pending.

§ 05

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 Guardian 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. Guardian's position is that each company and individual are responsible for their own due diligence and continuous security. Guardian'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 Guardian is 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, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

\$45B+DIGITAL ASSETS UNDER
REVIEW**200+**

ENGAGEMENTS COMPLETED

500+CRITICAL & HIGH FINDINGS
REPORTED

Reach us at audits@guardianaudits.com.