macro

Kwenta A-5

Security Audit

June 22, 2023
Version 1.0.0

Presented by 0xMacro

Table of Contents

Introduction

This document includes the results of the security audit for Kwenta's smart contract code as found in the section titled ‘Source Code’. The security audit was performed by the Macro security team from June 19, 2023 to June 20, 2023.

The purpose of this audit is to review the source code of certain Kwenta Solidity contracts, and provide feedback on the design, architecture, and quality of the source code with an emphasis on validating the correctness and security of the software in its entirety.

Disclaimer: While Macro’s review is comprehensive and has surfaced some changes that should be made to the source code, this audit should not solely be relied upon for security, as no single audit is guaranteed to catch all possible bugs.

Overall Assessment

The following is an aggregation of issues found by the Macro Audit team:

Severity	Count	Acknowledged	Won't Do	Addressed
Code Quality	1	-	-	1
Informational	3	3	-	-

Kwenta was quick to respond to these issues.

Specification

Our understanding of the specification was based on the following sources:

- Discussions on Discord with the Kwenta team.
- KIP-18
- KIP-19
- KIP-55

Trust Model, Assumptions, and Accepted Risks (TMAAR)

- Kwenta accounts use an upgradable beacon proxy, allowing the owner of the factory contract, a protocol DAO, to upgrade all accounts to a new implementation as well as pause account execution. It also has the ability to permanently turn off its ability to upgrade.
- Kwenta uses Gelato to execute automated conditional orders on behalf of users. There is an assumption that Gelato's executors will execute tasks set by users when the conditions are met, and only then. Gelato's executors currently require a 150k GEL token stake, KYB and DAO approval to join, and are subject to slashing penalties on their stake for acting maliciously.

Source Code

The following source code was reviewed during the audit:

- **Repository:** margin-manager
- **Commit Hash:** `c8c243770c9b7582700a1072646d4d28166bbecf`

Specifically, we audited the following contracts within this repository:

Contract	SHA256
src/Account.sol	`1109dc8872b6c463f205df29aa41aeb5322f69b803041479643b5307ee725498`
src/Events.sol	`0cb8460251b262fdb87eda10f39f6e7b27dbc03f2be603393ee0c6aec4af108e`

Note: This document contains an audit solely of the Solidity contracts listed above. Specifically, the audit pertains only to the contracts themselves, and does not pertain to any other programs or scripts, including deployment scripts.

Issue Descriptions and Recommendations

Click on an issue to jump to it, or scroll down to see them all.

Security Level Reference

We quantify issues in three parts:

1. The high/medium/low/spec-breaking **impact** of the issue:
    - How bad things can get (for a vulnerability)
    - The significance of an improvement (for a code quality issue)
    - The amount of gas saved (for a gas optimization)
2. The high/medium/low **likelihood** of the issue:
    - How likely is the issue to occur (for a vulnerability)
3. The overall critical/high/medium/low **severity** of the issue.

This third part – the severity level – is a summary of how much consideration the client should give to fixing the issue. We assign severity according to the table of guidelines below:

Severity	Description
(C-x) Critical	We recommend the client **must** fix the issue, no matter what, because not fixing would mean **significant funds/assets WILL be lost.**
(H-x) High	We recommend the client **must** address the issue, no matter what, because not fixing would be very bad, *or* some funds/assets will be lost, *or* the code's behavior is against the provided spec.
(M-x) Medium	We recommend the client to **seriously consider** fixing the issue, as the implications of not fixing the issue are severe enough to impact the project significantly, albiet not in an existential manner.
(L-x) Low	The risk is small, unlikely, or may not relevant to the project in a meaningful way. Whether or not the project wants to develop a fix is up to the goals and needs of the project.
(Q-x) Code Quality	The issue identified does not pose any obvious risk, but fixing could improve overall code quality, on-chain composability, developer ergonomics, or even certain aspects of protocol design.
(I-x) Informational	Warnings and things to keep in mind when operating the protocol. No immediate action required.
(G-x) Gas Optimizations	The presented optimization suggestion would save an amount of gas significant enough, in our opinion, to be worth the development cost of implementing it.

Issue Details

~~Q-1~~ Inaccurate Comment

TOPIC	STATUS	QUALITY IMPACT
Code Quality	Fixed	Low

In Account.sol, there is a comment above its constructor that mentions disabling initializers.

```
/// @notice disable initializers on initial contract deployment
```

Reference: Account.sol#L102

However, there are no initializers in this contract to disable, and this comment was left over from when Account.sol had been initialized prior.

Remediations to Consider

Remove this comment, as it is no longer accurate.

I-1 Conditional orders can be executed without set conditions met

TOPIC	STATUS	IMPACT
Informational	Acknowledged	Informational ✳

The `checker()` function called by Gelato executors to determine if a conditional order should be executed is never verified on-chain and these conditional orders can be called by any executor. This allows a rogue executor to execute conditional orders set by users without the set condition being

true. This is unlikely to occur, as the executor will have their GEL token stake slashed, but is something to be aware of if setting large conditional orders.

I-2 Conditional order execution may be delayed

TOPIC	STATUS	IMPACT
Informational	Acknowledged	Informational ✳

There is no guarantee that a conditional order will execute as soon as its condition is met, an executor may not execute the order while assigned to the task, or it may be executed with a delay. Executors are disincentivized to not execute valid orders, but if an executor doesn't, then execution may be delayed until a new executor is assigned to the task. It is suggested that conditional orders used to prevent liquidation should assume that the order could be delayed, and set a generous buffer to the targetPrice to reduce the likelihood of liquidation if there is a delay in execution.

I-3 ETH in an account can be griefed by an executor

TOPIC	STATUS	IMPACT
Informational	Acknowledged	Informational ✳

The fee paid to a Gelato executor for executing a conditional order is set by the executor themselves. If ETH is stored in an account, typically to pay for Gelato conditional orders, and there is an active conditional order, there is a chance for a malicious Gelato executor to set the fee to execute the order to be an amount equal to the ETH in the account. This allows all ETH in the account to be sent to the Gelato contract, and would require trust that the Gelato DAO would distribute this ETH back to the account at a later time. This is unlikely to occur as the executor would be disincentivized to do so as their stake would be slashed. It is recommended that users be aware of this possibility and not overly fund their accounts with ETH.

Disclaimer

Macro makes no warranties, either express, implied, statutory, or otherwise, with respect to the services or deliverables provided in this report, and Macro specifically disclaims all implied warranties of merchantability, fitness for a particular purpose, noninfringement and those arising from a course of dealing, usage or trade with respect thereto, and all such warranties are hereby excluded to the fullest extent permitted by law.

Macro will not be liable for any lost profits, business, contracts, revenue, goodwill, production, anticipated savings, loss of data, or costs of procurement of substitute goods or services or for any claim or demand by any other party. In no event will Macro be liable for consequential, incidental, special, indirect, or exemplary damages arising out of this agreement or any work statement, however caused and (to the fullest extent permitted by law) under any theory of liability (including negligence), even if Macro has been advised of the possibility of such damages.

The scope of this report and review is limited to a review of only the code presented by the Emergent team and only the source code Macro notes as being within the scope of Macro's review within this report. This report does not include an audit of the deployment scripts used to deploy the Solidity contracts in the repository corresponding to this audit. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. In this report you may through hypertext or other computer links, gain access to websites operated by persons other than Macro. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such websites' owners. You agree that Macro is not responsible for the content or operation of such websites, and that Macro shall have no liability to your or any other person or entity for the use of third party websites. Macro assumes no responsibility for the use of third party software and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.