macro

Clanker A-1

Security Audit

March 18, 2025
Version 1.0.0

Presented by 0xMacro

Table of Contents

Introduction

This document includes the results of the security audit for Clanker's smart contract code as found in the section titled ‘Source Code’. The security audit was performed by the Macro security team from March 11, 2025 to March 17, 2025.

The purpose of this audit is to review the source code of certain Clanker 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
Low	2	1	-	1
Code Quality	4	-	-	4
Informational	1	-	-	-

Clanker was quick to respond to these issues.

Specification

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

- Discussions on Telegram with the Clanker team.

Source Code

The following source code was reviewed during the audit:

- **Repository:** contracts
- **Commit Hash:** `0805bb89bceaf39a9a1e6422a61dfefce3dddcc1`

Specifically, we audited the following contracts within the repository:

Source Code	SHA256
src/Clanker.sol	ac3555b7232bf007dcc67a656a645ac5006656ab5055079570c0f266d2b42816
src/ClankerDeployer.sol	8d346d9e8192afd41ab57f65d42514379b4263b2722b790ceca0e0f4c24a2d6f
src/ClankerToken.sol	e86210da783df250cb3a3de1343e967ade379359621f0ec0868eb4a35774a132
src/ClankerVault.sol	97eb8c784315b6efa99e5012addefa0a01a3d7eb797d275a8dc18a81e5cc4dbc
src/LpLockerv2.sol	115eb7ceea6014642f0670f5e848312cb1e6d2d54eb0ad04dca6403bd56195c2
src/interfaces/IClanker.sol	cf78b237444395e9fef8b4be69e4f1f3bdfe2945be1ecb974f2895e235947acc
src/interfaces/IClankerVault.sol	e89eac07d18cae598dfe184b57078a3d76c5aeae14f1a13778417ee147e35314
src/interfaces/ILPLockerv2.sol	fa57253d70478b722e4e5e549c2337332267e379c19b4065c8ff263565501a31
src/interfaces/uniswapv3.sol	8aacdba548c8fc33d871a9c382546cf26e3ade99d28f45aaa0f0153677481acd

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

L-1 Blacklisted creatorRecipient or interfaceRecipient may prevent reward collection

TOPIC	STATUS	IMPACT	LIKELIHOOD
Spec	Acknowledged	Low	Low

In the `LpLockerv2.sol` contract, the `collectRewards()` function retrieves liquidity provider rewards, calculates the proportional share of rewards for the creator, interface, and team, and distributes this to each actor. However, if any of these actors is blacklisted on reward tokens that support blacklisting (e.g., USDC), the corresponding token transfer would fail. As a result, none of the actors can collect rewards.

If actors are non-malicious, the team, creator, or interface may update their recipient address and unblock the reward collection. Otherwise, reward collection will be permanently blocked if the actor cannot access the admin address to invoke the recipient update operation or is unwilling to do it for other reasons.

Remediations to Consider

- Update reward collection implementation to use assets pull instead of assets push pattern.

RESPONSE BY CLANKER

> The team decided this was an edge case as the majority of our pairs are not in USDC/USDT. We will plan a fix in our next release.

~~L-2~~ Unexpected reward distribution in case of drift of constants defining reward share

TOPIC	STATUS	IMPACT	LIKELIHOOD
Spec	Fixed	Low	Low

In the `LpLockerv2.sol` contract, the `collectRewards()` function calculates the team's reward based on `TEAM_REWARD` constant defined in `LpLockerv2.sol`. On the other hand, `MAX_CREATOR_REWARD` constant is set in the constructor based on the value of `Clanker.MAX_CREATOR_REWARD` constant. In case when `MAX_CREATOR_REWARD` is 80, and the corresponding creator reward is equal to `MAX_CREATOR_REWARD`, the current implementation in `collectRewards()` would distribute shares as expected.

However, if `MAX_CREATOR_REWARD` in Clanker is set to 70 and the corresponding creator reward remains equal to `MAX_CREATOR_REWARD` (also 70 in this case), the current implementation in `collectRewards()` would distribute rewards in unexpected amounts. For example, in this specific case, the share of the reward for the team would remain at 20%, but the creator share would be 80% and not 70%, as defined by the `MAX_CREATOR_REWARD` constant value.

Remediations to Consider

- Add validation to `LpLockerv2.sol` constructor to verify that the sum of `MAX_CREATOR_REWARD` and the `TEAM_REWARD` is 100%.

~~Q-1~~ ClankerToken does not indicate interface support for all implemented capabilities

TOPIC	STATUS	QUALITY IMPACT

Best practices	Fixed	Low

In the `ClankerToken.sol` contract, `supportsInterface()` is implemented in the following way and indicates support for IERC7802, IERC20, and IERC165.

```
function supportsInterface(bytes4 _interfaceId) public pure returns (bool) {
    return _interfaceId == type(IERC7802).interfaceId
        || _interfaceId == type(IERC20).interfaceId || _interfaceId == type(IER
}
```

However, as `ClankerToken` inherits `ERC20Votes`, it has additional capabilities represented by the `IERC5805` (IVotes and IERC6372) interface, which are not properly indicated as supported.

Remediations to Consider

- Consider updating `supportsInterface()` implementation to indicate `IERC5805` support.

~~Q-2~~ Missing events for important state changes

TOPIC	STATUS	QUALITY IMPACT
Events	Fixed	Low

Missing events for important state changes

- In `Clanker.sol`, consider adding events for `setDeprecated()` and `setAdmin()` functions.

- In `ClankerToken.sol`, consider adding events for `updateImage()` and `updateMetadata()`

~~Q-3~~ Missing validation for default team recipient

TOPIC	STATUS	QUALITY IMPACT
Events	Fixed	Low

In `LpLockerv2.sol`, the `teamRecipient` storage variable contains an address where rewards for the team would be sent. In addition, this address is the default recipient of creator and interface reward shares if they are not set.

Reward assets can be permanently lost if the `teamRecipient` value is incorrectly set or updated. Therefore, consider adding `teamRecipient` value validation.

~~Q-4~~ Inconsistent update events

TOPIC	STATUS	QUALITY IMPACT
Events	Fixed	Low

The codebase contains inconsistencies in defining events that treat storage variable updates. Sometimes, only a new value is emitted, e.g., `LpLockerv2.TeamRecipientUpdated`. On the other hand, in the events such as `ClankerVault.MinimumVaultTimeUpdated` or `ClankerVault.AllocationAdminUpdated`, both old and new, values are emitted.

Consider updating event definitions to have more consistent implementation and to facilitate easier off-chain tracking and monitoring integration.

I-1 Admin may reconfigure the Clanker system dynamically

TOPIC	IMPACT
Trust model	Informational ✳

In the `Clanker` contract, the `initialize()` function is accessible only by the owner. This function allows the owner to configure references to important underlying components, such as ClankerVault, LpLockerv2, Uniswap V3 Factory, Position Manager, and Swap Router.

The owner may invoke `initialize()` multiple times, reconfiguring the Clanker system with different underlying system contracts. As a result, the system is not immutable, and due to potential changes, it requires additional user trust.

To minimize required user trust, consider not allowing multiple `initialize()` invocations or adding a feature to disable multiple invocations when the system becomes stable.

RESPONSE BY CLANKER

> The team decided that we want the ability to reconfigure if needed due to the complexity of multichain deployments. If the owner keys get leaked to a malicious other party, we'd prefer our users to stop using the affected contracts as we're no longer connected to them.

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 Clanker 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.