Rogue Finance Security Review

Version 1.1

September 20, 2023

Conducted by:
Georgi Georgiev (Gogo), Independent Security Researcher

Table of Contents

1 About Gogo

Georgi Georgiev, known as Gogo, is an independent security researcher specialized in Solidity smart contracts auditing and bug hunting. Having conducted numerous solo and team smart contract security reviews, he always strives to deliver top-quality security auditing services. For security consulting, you can contact him on Twitter, Telegram, or Discord - *@gogotheauditor*.

2 Disclaimer

Audits are a time, resource and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to find as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

4 Executive summary

Overview

Project Name	Rogue Finance
Repository	https://github.com/scamilcar/locker
Commit hash	e2035fa7cd1b95cd630e50880ff1f6fa6ecdd026
Resolution	037bb120849afdfa4617a4a440b1f4cc9a9d8f7d
Documentation	https://roguefinance.gitbook.io
Methods	Manual review

Scope

src/Locker.sol

Issues Found

Critical risk	0
High risk	0
Medium risk	1
Low risk	0
Informational	3
Gas optimization	4

5 Findings

5.1 Medium risk

5.1.1 Admin can steal all MAV tokens before the Board contract is set

Severity: *Medium risk*

Context: Locker.sol#L88, Locker.sol#L104

Description: The `Locker` is an ERC20 contract that mints tokens (rMAV) to depositors of the MAV token. The deposited assets are then used to stake into Maverick's voting-escrow contract (veMAV). This is done via the `lock` function, which makes the following external call:

```
function lock() external {
    if (!disabled) revert NotDisabled();
    ...
    IBoard(board).extendLockup(balance);
}
```

The `Board` contract is out of scope for this audit as it is not fully implemented yet. For the same reason, the `Locker` contract implements a function that allows the contract owner to set the `board` address (only once) after the `Board` contract development has finished and the contract has been deployed.

However, since the `board` address variable can be set after users have deposited their MAV tokens into the `Locker` and also eventually after the withdrawals have been disabled, the contract owner can set any arbitrary contract address as the `board` and then immediately steal the deposited MAV tokens:

```
function setBoard(address _board, uint256 _callIncentive) external onlyOwner {
    ...
    board = _board;
    mav.safeApprove(_board, type(uint256).max); // @audit Can be malicious contract
}
```

This would result in a full loss of all deposited MAV tokens and eventually making the minted rMAV tokens (that depositors have received) worthless as they rely on the project's value.

Recommendation: Use a timelock when setting the `board` contract to allow users to withdraw their tokens in a specific time window in case a malicious or just wrong `board` address has been set:

```
uint256 boardProposedAt; // @audit Add a timestamp variable.

function setBoard(address _board, uint256 _callIncentive) external onlyOwner {
    ...
    boardSetAt = block.timestamp; // @audit Update when set by admin.
    ...
}

function disable() external onlyOwner {
    // @audit Disable withdrawals and enable locks only after a 3-day period
    // so that users have time to review the proposal and withdraw if needed.
    if (block.timestamp < boardSetAt + 3 days || boardSetAt == 0) revert
        TimelockPeriodNotPassed();
    ...
}
```

Resolution: Resolved. The recommended fix was implemented and reviewed at commit 037bb12.

5.2 Informational

5.2.1 Missing event emission

Severity: *Informational*

Context: Locker.sol#L113, Locker.sol#L98-L106

Description: Both the `setBoard` and `updateIncentive` functions change the value of the `callIncentive` variable. However, only the `updateIncentive` function emits the `IncentiveUpdated(_callIncentive)` event.

Recommendation: Emit `IncentiveUpdated` in the `setBoard` function.

Resolution: Resolved. The client added an `incentive` parameter to the already emitted `BoardSet` event.

5.2.2 Implementation differs from documentation requirements

Severity: *Informational*

Context: Locker.sol#L63

Description: The documentation states that "Depositors receive an ERC20 (rMAV) at a 1:1 ratio". However, the `deposit` function pulls funds from the msg.sender and then mints the rMAV tokens to the passed `recipient` address instead of the depositor.

Recommendation: Consider either correcting the documentation or the `deposit` function implementation.

Resolution: Resolved. The documentation was corrected, depositors can pass a `recipient` address that will receive the minted shares.

5.2.3 CEI pattern not adhered to

Severity: *Informational*

Context: Locker.sol#L88-L89

Description: The Checks-Effects-Interactions pattern is not completely followed in `lock` where the external call is performed before the `incentive` tokens have been minted to the caller:

```
IBoard(board).extendLockup(balance);
_mint(msg.sender, incentive);
```

Recommendation: Execute the external call after the `_mint`.

Resolution: Resolved. The recommendation was implemented and reviewed at commit 7266f76.

5.3 Gas Optimization

5.3.1 Using Math.mulDiv incurs gas overhead

Severity: *Gas Optimization*

Context: Locker.sol#L87

Description: The caller's call incentive amount of rMAV tokens is calculated in `lock` with the following line:

```
uint256 incentive = Math.mulDiv(balance, callIncentive, ONE);
```

Simply using `balance * callIncentive / ONE` is cheaper because the additional logic that the `mulDiv` method implements is not needed in this case.

Recommendation: Change the aforementioned line to:

```
uint256 incentive = balance * callIncentive / ONE;
```

Resolution: Resolved.

5.3.2 Tracking the total locked MAV tokens in storage is redundant

Severity: *Gas Optimization*

Context: Locker.sol#L33

Description: The `totalLocked` variable is used to track the total amount of MAV tokens deposited into the Rogue Protocol. It is increased on deposits and decreased on withdrawals. However, it is never used for any other logic, making it a waste of gas.

Recommendation: Remove the `totalLocked` variable.

Resolution: Resolved.

5.3.3 Redundant balance validation check when withdrawing deposited tokens

Severity: *Gas Optimization*

Context: Locker.sol#L75

Description: The following check in the `withdraw` function can be removed, as `_burn` would execute it anyways:

```
function withdraw(uint256 amount) external {
    ...
    if (balanceOf(msg.sender) < amount) revert LowBalance(); // @audit Redundant
    _burn(msg.sender, amount);
```

Recommendation: Remove the above check.

Resolution: Resolved.

5.3.4 Redundant storage address validation check when extending veMAV staking lock

Severity: *Gas Optimization*

Context: Locker.sol#L84

Description: The following check in the `lock` function can be removed, as the external call made later will revert if the condition is true anyway:

```
function lock() external {
    ...
    if (board == address(0)) revert BoardNotSet(); // @audit Redundant
    ...
    IBoard(board).extendLockup(balance);
```

Recommendation: Remove the above check.

Resolution: Resolved.