Maverick V2 Rewards Security Review

Report Version 1.0

May 20, 2024

Conducted by:

George Hunter, Independent Security Researcher

Table of Contents

1 About George Hunter

George Hunter is a proficient and reputable independent smart contract security researcher with over 100 solo and team security engagements contributing to the security of numerous smart contract protocols in the past 2 years. Previously held roles include Lead Smart Contract Auditor at Paladin Blockchain Security and Smart Contract Engineer at Nexo. He has audited smart contracts for clients such as LayerZero, Euler, TraderJoe, Maverick, Ambire, and other leading protocols. For security audit inquiries, you can reach out on Telegram or Twitter at *@georgehntr*.

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 identify as many vulnerabilities as possible. Audits can reveal the presence of vulnerabilities **but cannot guarantee their absence**.

3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	High	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** - involves a small loss of funds or affects a core functionality of the protocol.
- **Low** - encompasses any unexpected behavior that is non-critical.

3.2 Likelihood

- **High** - a direct attack vector; the cost is relatively low compared to the potential loss of funds.
- **Medium** - only a conditionally incentivized attack vector, with a moderate likelihood.
- **Low** - involves too many or unlikely assumptions; offers little to no incentive.

3.3 Actions required by severity level

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

4 Executive summary

George Hunter was engaged by Maverick to review the Maverick V2 Rewards smart contracts.

Overview

Project Name	Maverick V2 Rewards contracts
Repository	https://github.com/maverickprotocol/maverick-v2
Commit hash	912bf01365ffd87b4149f66bb96f8e5a816ca9a0
Resolution	07ad29f773f16bdfbae3d97d3a7c2f9d64866093
Methods	Manual review

Scope

v2-rewards/contracts/*

Issues Found

High risk	1
Medium risk	0
Low risk	0
Informational	1

5 Findings

5.1 High

5.1.1 Rewards at index 0 not updated upon stake and unstake

Severity: *High*

Context: MaverickV2Reward.sol#L435

Description: The *_updateAllRewards* method in *MaverickV2Reward* is called before executing the *stake* and *unstake* operations in order to update the reward state associated with the given *tokenId*:

```
function _updateAllRewards(uint256 tokenId) internal {
    for (uint8 i = 1; i < rewardTokenCount; i++) {
        RewardData storage data = rewardData[i];

        _updateReward(tokenId, data);
    }
}
```

The problem is that the iteration starts at *i = 1* while it should be starting at *i = 0* to update the first reward element of the *rewardData* array.

Recommendation: Begin the loop at index = 0, instead of 1.

Resolution: Resolved.

5.2 Informational

5.2.1 Typographical mistakes, code-style suggestions and non-critical issues

Severity: *Informational*

Context: MaverickV2Reward.sol#L70, VotingEscrow.sol#L267-L283

Description: The contracts contain one or more typographical issues, code-style suggestions and non-critical issues. In an effort to keep the report size reasonable, we enumerate these below:

- *_rewardGetter* in *MaverickV2Reward* is never used.
- The *approve* method can also be overwritten to revert in VotingEscrow.

Recommendation: Consider fixing the above typographical issues, code-style suggestions and non-critical issues.

Resolution: Resolved.