Megaeth: MegaSaleRefund

Security Review

Solo review by:
Chinmay Farkya, Security Researcher

December 4, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

A security review is a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While the review endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that a security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings are a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

MegaETH is a high-performance blockchain network (currently in testnet) that is EVM-compatible and designed for ultra-fast block times and real-time applications.

From Nov 27th to Nov 28th the security researchers conducted a review of MegaETH - MegaSaleRefund on commit hash 409d3de5. A total of **3** issues were identified:

Issues Found

Severity	**Count**	**Fixed**	**Acknowledged**
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	0	0	0
Gas Optimizations	1	1	0
Informational	2	1	1
Total	**3**	**2**	**1**

2.1 Scope

The security review had the following components in scope for MegaETH - MegaSaleRefund on commit hash 409d3de5:

```
src/MegaSaleRefund.sol
```

3 Findings

3.1 Gas Optimization

3.1.1 `batchRefund()` logic can be simplified

Severity: Gas Optimization

Context: *(No context files were provided by the reviewer)*

Description: The `batchRefund()` function emits an event

```
BatchRefundProcessed(uint256 usersCount, uint256 totalAmount);
```

and makes use of the following logic to track `usersCount` and `totalAmount` refunded in the batch:

```
uint256 batchTotalAmount = 0;
uint256 successfulRefunds = 0; // for tracking user count

for (uint256 i = 0; i < usersCount; i++) {
    _processSingleRefund(users[i], amounts[i], merkleProofs[i]);
    batchTotalAmount += amounts[i];
    successfulRefunds++;
}
```

But this local variable `successfulRefunds` is not required. The `batchRefund` only ever executes in full, it succeeds only if all refunds succeed as per the input, and we already have the user count from the input `users` array.

Recommendation: Consider using the already known users count for event data, and removing the `successfulRefunds` variable.

MegaETH: Fixed in commit ab337504.

Chinmay Farkya: Fix verified.

3.2 Informational

3.2.1 Incorrect natspec

Severity: Informational

Context: MegaSaleRefund.sol#L160

Description: The natspec documentation above `_processSingleRefund()` function says:

```
// PREREQUISITE: sale contract must grant TOKEN_RECOVERER_ROLE to this contract
// @dev This is checked in _processRefund() via sale.hasRole(sale.TOKEN_RECOVERER_ROLE(),
 ↪  address(this))
```

But this is an incorrect description. The mentioned check is actually done in `refund()` and `batchRefund()` functions, and `_processRefund()` function does not even exist.

Recommendation: Consider changing the natspec comments.

MegaETH: Fixed in commit 6bdfba6c.

Chinmay Farkya: Fix verified.

3.2.2 Make sure that `TokenSale` entity addresses can handle refunded USDT

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The `TokenSale` contract provided a way for users to bid with USDT, and the `MegaSaleRefund` contract is aimed at providing 10% rebates for users who opted in to lock the $MEGA tokens for a year.

The `refund()` function here uses the input `to address` to check user entry in the merkle tree (list of all user entities that are eligible for refund), and also sends the refunded USDT back to this ”to” address. The user does not have a way to receive the refunds at a different address.

This works fine if all entities that interacted with `TokenSale` contract were EOAs. But if they were contracts with hardcoded pathways, they might not be able to pull those USDT out after receiving the refund at their own contract.

Recommendation: Consider checking if user entities would have a problem handling received USDT refunds, if they used smart contracts for the public sale.

MegaETH: Acknowledged.

Chinmay Farkya: Acknowledged.