

S A L U S S E C U R I T Y

O C T 2 0 2 5



CODE SECURITY ASSESSMENT

M E T A A R E A N

Overview

Project Summary

- Name: MetaArena - Token
- Platform: EVM-compatible chains
- Language: Solidity
- Address:
 - [0xAaFe1f781bC5e4D240c4B73f6748d76079678Fa8](#)
- Audit Range: See [Appendix - 1](#)

Project Dashboard

Application Summary

Name	MetaArena - Token
Version	v2
Type	Solidity
Dates	Oct 15 2025
Logs	Oct 15 2025; Oct 15 2025

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	3
Total	4

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction	4
1.1 About SALUS	4
1.2 Audit Breakdown	4
1.3 Disclaimer	4
Findings	5
2.1 Summary of Findings	5
2.2 Notable Findings	6
1. Centralization risk with initial token distribution	6
2.3 Informational Findings	7
2. Lack of Security Contact Information for Responsible Disclosure	7
3. Lack of License Specification	8
4. Literal Number Safety	9
Appendix	10
Appendix 1 - Files in Scope	10

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (<https://t.me/salusec>), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Centralization risk with initial token distribution	Low	Centralization	Resolved
2	Lack of Security Contact Information for Responsible Disclosure	Informational	Configuration	Acknowledged
3	Lack of License Specification	Informational	Configuration	Acknowledged
4	Literal Number Safety	Informational	Configuration	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk with initial token distribution

Severity: Low

Category: Centralization

Target:

- TIMI.sol

Description

contracts/TIMI.sol:L33 - L48

```
constructor() ERC20("TIMI", "TIMI") {
    _mint(msg.sender, 2100000000 ether);
}
```

When the contract is deployed, `\$TIMI` is sent to one account. This account then has full control over the token distribution. If it is an EOA account, any compromise of its private key could drastically affect the project – for example, attackers could manipulate the price of `\$TIMI` on the DEX if they gain access to the private key.

Recommendation

It is recommended to transfer tokens to a multi-sig account and promote transparency by providing a breakdown of the intended initial token distribution in a public location.

Status

The team has resolved this issue by transferring the token-holding address from an EOA to a multi-sig address.

2.3 Informational Findings

2. Lack of Security Contact Information for Responsible Disclosure

Severity: Informational

Category: Configuration

Target:

- TIMI.sol

Description

The contract `TIMI` does not specify a security contact point. Including a designated security contact (such as an email address or ENS name) in the contract's NatSpec header facilitates responsible vulnerability disclosure. This makes it easier for external researchers to quickly reach the appropriate team in the event a vulnerability is identified, helping minimize the time window between discovery and mitigation. The Ethereum community has begun standardizing this practice using the `@custom:security-contact` tag, adopted by tools such as OpenZeppelin Wizard and ethereum-lists.

Recommendation

Consider adding a NatSpec comment at the top of the contract with a `@custom:security-contact` field pointing to the preferred disclosure channel.

Status

This issue has been acknowledged by the team.

3. Lack of License Specification

Severity: Informational

Category: Configuration

Target:

- TIMI.sol

Description

The contract `TIMI` does not specify a license or specify an incorrect one (e.g., // SPDX-License-Identifier: UNLICENSED).

Recommendation

It is recommended to explicitly specify a license (e.g., MIT or GPL-3.0) in all contracts and interfaces and maintain consistency across the codebase. This establishes clear legal usage terms, prevents misuse, and encourages collaboration.

Status

This issue has been acknowledged by the team.

4. Literal Number Safety

Severity: Informational

Category: Configuration

Target:

- TIMI.sol

Description

Throughout the codebase, an instance of literal numbers being used directly were identified:

```
constructor() ERC20("TIMI", "TIMI") {
    _mint(msg.sender, 2100000000 ether);
}
```

Direct use of literal values with so many digits impairs code readability and may lead to unexpected consequences.

Recommendation

For multi-digit literal numbers, it is recommended to use the [Ether Suffix](#), [Time Suffix](#), or [Scientific Notation](#).

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files at address

[0xAaFe1f781bC5e4D240c4B73f6748d76079678Fa8:](#)

File	SHA-1 hash
TIMI.sol	7db75e626ca2031c0755d4723ff0f13ae2972195