SALUS SECURITY

MAY 2023

CODE SECURITY ASSESSMENT

METAMERGE

Overview

Project Summary

- Name: MetaMerge - MetaMergeMana token
- Version: commit 2c6992b
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository: https://github.com/MetaMergeXYZ/Mana
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	MetaMerge - MetaMergeMana token
Version	v2
Type	Solidity
Dates	May 11 2023
Logs	May 10 2023; May 11 2023

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

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	Low	Centralization	Acknowledged
2	Functions visibility	Informational	Gas inefficiencies	Acknowledged
3	Missing address validation	Informational	Validation	Acknowledged
4	Floating compiler version	Informational	Configuration	Acknowledged

2.2 Notable Findings

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

1. Centralization risk	
Severity: Low	Category: Centralization
Target: - src/Mana/MetaMergeMana.sol	

Description

There are three privileged roles in the MetaMergeMana contract:

- the MINTER_ROLE can mint tokens to an address using the mint() function.
- the PAUSER_ROLE can pause and unpause the contract. When paused, token transfers is disabled, which also disables token minting and burning.
- the DEFAULT_ADMIN_ROLE has the permission to grant and revoke the above roles.

If these privileged accounts are plain EOA accounts, this poses a risk to the users. If any of the private keys is compromised, an attacker could exploit the privileged operations to attack the project.

Recommendation

We recommend transferring privileged accounts to multi-signature accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

2. Functions visibility	
Severity: Informational	Category: Gas inefficiencies
Target: - src/Mana/MetaMergeMana.sol	

Description

For certain functions that are only called from outside the smart contract, using function visibility public can make the gas costs greater than using the visibility external. This is because with public functions, the EVM copies inputs (especially dynamic-sized arrays) into memory, while it reads from calldata if the function is external, which is cheaper.

In the MetaMergeMana contract, the mint(), pause(), and unpause() functions have public visibility whilst only being called externally.

Recommendation

Consider changing the visibility of public functions to external if they are only accessed externally.

Status

This issue has been acknowledged by the team.

3. Missing address validation

Severity: Informational	Category: Validation
Target: - src/Mana/MetaMergeMana.sol	

Description

To ensure security, it is best practice to check input addresses against zero-address, otherwise contract functionality may become inaccessible.

However, MetaMergeMana's constructor function lacks such a check for the admin_ parameter. If zero-address is mistakenly passed in as admin_, the DEFAULT_ADMIN_ROLE, MINTER_ROLE, and PAUSER_ROLE cannot be set, rendering the contract unusable.

Recommendation

Consider implementing a zero-address check for admin_ in the constructor.

Example:

```
require(admin_ != address(0), "MetaMergeMana: zero address");
```

Status

This issue has been acknowledged by the team.

4. Floating compiler version

Severity: Informational	Category: Configuration
Target: - src/Mana/MetaMergeMana.sol	

Description

```
pragma solidity ^0.8.18;
```

The MetaMergeMana contract uses a floating Solidity compiler version, ^0.8.18.

It's best to deploy contracts with the same compiler version and flags that they have been thoroughly tested with. Locking the compiler version helps to prevent contracts from being accidentally deployed using an outdated compiler version, which could introduce bugs that have a negative impact on the system.

Recommendation

It is recommended to use a locked Solidity compiler version.

Example:

```
pragma solidity 0.8.18;
```

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file in commit 2c6992b:

File	SHA-1 hash
src/Mana/MetaMergeMana.sol	d1378d3a2f9c59778570849d847618b220c9f0c2