SALUS SECURITY

NOV 2024

CODE SECURITY ASSESSMENT

WABA

Overview

Project Summary

- Name: WABA - MinerPool
- Platform: WABA chain
- Language: Solidity
- Address: 0x9719C8c0e952DD7142A7858e0d493b43D063CA92
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	WABA - MinerPool
Version	v2
Type	Solidity
Dates	Nov 12 2024
Logs	Oct 09 2024; Nov 12 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	2
Total informational issues	1
Total	3

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	Implementation contract could be initialized by everyone	Low	Business Logic	Acknowledged
2	The setRewardPerBlock() function unusable when contract is paused	Low	Business Logic	Acknowledged
3	Missing events for functions that change critical state	Informational	Logging	Acknowledged

2.2 Notable Findings

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

1. Implementation contract could be initialized by everyone

Severity: Low	Category: Business Logic
Target: - MinerPool.sol	

Description

According to OpenZeppelin, the implementation contract should not be left uninitialized.

An uninitialized implementation contract can be taken over by an attacker, which may impact the proxy. There is nothing preventing the attacker from calling the `initialize()` function in MinerPool's implementation contract.

Recommendation

To prevent the implementation contract from being used, consider invoking the _disableInitializers function in the constructor of the `MinerPool` contract to automatically lock it when it is deployed.

Status

This issue has been acknowledged by the team.

2. The setRewardPerBlock() function unusable when contract is paused

Severity: Low	Category: Business Logic
Target: - MinerPool.sol	

Description

The setRewardPerBlock function in the MinerPool contract cannot be executed when the contract is in a paused state due to its dependency on the updatePool function, which has a whenNotPaused modifier. This design flaw prevents the owner from adjusting the reward rate during a pause, leading to numerous significant issues.

MinerPool.sol:L767-L770

```
function setRewardPerBlock(uint256 reward) external onlyOwner {
    updatePool();
    rewardPerBlock = reward;
}

function updatePool() public whenNotPaused {
    ...
}
```

This architecture creates several problems:

- Economic Inflexibility: Inability to adjust rewards quickly in response to market volatility, potentially leading to token inflation or miner disengagement.
- Extended Vulnerability Windows: Security issues requiring both a pause and reward adjustment cannot be addressed simultaneously, potentially extending the exposure to risks.
- Strategic Manipulation: Knowledgeable actors could exploit the predictable behavior around pauses.

Recommendation

Consider implementing the following change, same as in adminClaim:

```
if (!paused()) {
    updatePool();
}
```

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

3. Missing events for functions that change critical state

Severity: Informational	Category: Logging
Target: - MinerPool.sol	

Description

Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes that allow users to evaluate them. Missing events do not promote transparency and if such changes immediately affect users' perception of fairness or trustworthiness, they could exit the protocol causing a reduction in protocol users.

In the MinerPool, events are missing in the following functions:

- grantRole()
- revokeRole()
- setRewardPerBlock()

Recommendation

It is recommended to emit events for critical state changes.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered files at the following address:

0x9719C8c0e952DD7142A7858e0d493b43D063CA92: