SALUS SECURITY

DEC 2023

CODE SECURITY ASSESSMENT

BITUNEAI

Overview

Project Summary

- Name: BituneAi
- Platform: Ethereum
- Address: 0xc203757eCa55f18aF6aFDf3967fE042a707CD8bA
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	BituneAi
Version	v2
Type	Solidity
Dates	Dec 05 2023
Logs	Dec 05 2023; Dec 05 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	5
Total	6

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	Resolved
2	Use of floating pragma	Informational	Configuration	Resolved
3	SafeMath is not required since Solidity 0.8.0	Informational	Redundancy	Resolved
4	Lack of indexed parameters in events	Informational	Logging	Resolved
5	Missing zero address checks	Informational	Data Validation	Resolved
6	Gas optimization suggestions	Informational	Gas Optimization	Resolved

2.2 Notable Findings

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

1. Centralization risk	
Severity: Low	Category: Centralization
Target: - BituneAi.sol	

Description

There is a privileged owner role in the BituneAi contract. The owner of the BituneAi contract can withdraw specified tokens from the contract.

BituneAi.sol:L2354-L2357

```
function withdrawToken(address _token,address _to,uint256 _amount) external {
    require(msg.sender == owner);
    IERC20(_token).safeTransfer(_to,_amount);
}
```

If the owner's private key is compromised, an attacker could withdraw any ERC20 tokens in the contract. If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

Recommendation

We recommend transferring privileged accounts to multi-sig 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 resolved by the team. The team has removed the owner role and the withdrawToken() function.

2.3 Informational Findings

2. Use of floating pragma	
Severity: Informational	Category: Configuration
Target: - BituneAi.sol	

Description

```
pragma solidity ^0.8.20;
```

The BituneAi contract uses a floating compiler version ^0.8.20.

Using a floating pragma ^0.8.20 statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been resolved by the team.

3. SafeMath is not required since Solidity 0.8.0	
Severity: Informational	Category: Redundancy
Target: - BituneAi.sol	

Description

The contract uses SafeMath library for mathematical operations to avoid overflow and underflow. But there are inbuilt overflow and underflow operations from solidity version 0.8.0. So, the use of the SafeMath library is redundant.

BituneAi.sol:L2339

```
using SafeMath for uint;
```

Recommendation

It is recommended to remove the SafeMath library.

Status

This issue has been resolved by the team.

4. Lack of indexed parameters in events	
Severity: Informational	Category: Logging
Target: - BituneAi.sol	

Description

In the BituneAi contract, there are no indexed event parameters.

BituneAi.sol:L2346

```
event Permutation(address account,uint256 amount);
```

Recommendation

Consider indexing important parameters, such as account, to improve off-chain services' ability to search and filter for specific events.

Status

This issue has been resolved by the team.

5. Missing zero address checks	
Severity: Informational	Category: Data Validation
Target: - BituneAi.sol	

Description

It is considered a security best practice to verify addresses against the zero address during initialization or setting. However, this precautionary step is absent for address variables owner and matterToken.

BituneAi.sol:L2349-L2350

```
constructor(address _token,string memory name, string memory symbol, uint256
totalSupply, address _owner) ERC20Permit(name) ERC20(name,symbol){
    matterToken = IERC20(_token);
    owner = _owner;
    ...
}
```

Recommendation

Consider adding zero address checks for address variables mentioned above.

Status

This issue has been resolved by the team.

6. Gas optimization suggestions	
Severity: Informational	Category: Gas Optimization
Target: - BituneAi.sol	

Description

BituneAi.sol:L2342-L2343

```
address public owner;
IERC20 public matterToken;
```

The variables above can be declared as immutable to save gas since their value can not be updated after contract deployment.

Recommendation

Consider applying the gas optimizations where needed.

Status

This issue has been resolved by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file provided by the client:

File	SHA-1 hash
BituneAi.sol	b93ee91d23ccc41e5d34fe87dc540f606c53bb98

The file and hash after revision are as follows:

File	SHA-1 hash
BituneAi.sol	93af0861d72c5bcba86a2d2198c36663dda48bab
Include2.sol	ddbdd529321feaf0a5f91a20da5c65a9581b40e0