SALUS SECURITY

FEB 2024

CODE SECURITY ASSESSMENT

BITSWAP

Overview

Project Summary

- Name: BitSwap - StakingRewards
- Platform: EVM-compatible chains
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	BitSwap - StakingRewards
Version	v2
Type	Solidity
Dates	Feb 06 2024
Logs	Feb 05 2024; Feb 06 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	3
Total informational issues	1
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	Missing events for functions that change critical state	Low	Logging	Acknowledged
2	Not using safeTransfer and safeTransferFrom	Low	Business Logic	Acknowledged
3	Centralization risk	Low	Centralization	Acknowledged
4	Not all EVM-compatible chains support Solidity 0.8.20	Informational	Configuration	Acknowledged

2.2 Notable Findings

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

1. Missing events for functions that change critical state	
Severity: Low	Category: Logging
Target: - StakingRewards.sol	

Description

Events capture changed parameters for off-chain tools/interfaces to register those changes. Lack of events hinders transparency.

The StakingRewards contract does not emit events for critical functions like stake(), withdraw(), getReward(), setRewardsDuration(), setProvider(), and notifyRewardAmount().

Recommendation

It is recommended to emit events for functions that change critical state.

Status

This issue has been acknowledged by the team.

2. Not using safeTransfer and safeTransferFrom

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

Description

StakingRewards.sol:L75, L84, L99

```
stakingToken.transferFrom(msg.sender, address(this), _amount);

stakingToken.transfer(msg.sender, _amount);

rewardsToken.transferFrom(provider, msg.sender, reward);
```

The StakingRewards contract uses transfer/transferFrom to transfer the stakingToken and rewardsToken without checking its return value. This can be problematic because certain tokens (e.g. ZRX, EURS) do not revert on failure, but instead return false. Consequently, if these tokens are used as stakingToken or rewardsToken, the contract may incorrectly assume that the transfer was successful.

Recommendation

Consider using safeTransfer and safeTransferFrom from the SafeERC20 library as a best practice.

Status

This issue has been acknowledged by the team.

3. Centralization risk

Severity: Low	Category: Centralization
Target: - StakingRewards.sol	

Description

The StakingRewards contract sets the deployer as the owner upon deployment and does not allow for changing the owner afterwards.

The owner has the ability to:

- Set the provider address
- Set the reward duration when reward distribution is finished
- Add reward amount by using notifyRewardAmount()

If the owner's private key is compromised, the attacker can set the provider address to address(0), temporarily preventing users from using getRewards().

Recommendation

It is recommended to use the Ownable2Step library for access control and transfer the owner role to a multi-sig account. Alternatively, deploy the StakingRewards contract using a multi-sig account as the owner. Using multi-sig accounts for privileged roles 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

4. Not all EVM-compatible chains support Solidity 0.8.20

Severity: Informational	Category: Configuration
Target: - StakingRewards.sol	

Description

Solidity version 0.8.20 introduced the support for PUSH0 opcode. Certain blockchains (e.g. Arbitrum) may not support this new feature, thereby causing the contract deployment to fail.

Recommendation

It is recommended to check if the solidity version you are using is supported by the target blockchain.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file provided by the client:

File	SHA-1 hash
StakingRewards.sol	61aba8cb4a8bcc4585e92c025a62fc1e1abfc252