SALUS SECURITY

MAY 2024

CODE SECURITY ASSESSMENT

BOUNCEBIT

Overview

Project Summary

- Name: BounceBit - LSD contract
- Platform: BounceBit Chain
- Language: Solidity
- Repository:
  - https://github.com/BounceBit-Labs/bouncebit-lsd-contract
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	BounceBit - LSD contract
Version	v2
Type	Solidity
Dates	May 29 2024
Logs	Apr 07 2024; May 29 2024

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	Mismatch between code and comment	Informational	Code Quality	Acknowledged
3	Could use a two-step ownership transfer	Informational	Business logic	Acknowledged
4	Missing zero address checks	Informational	Data Validation	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: - contracts/SystemConfig.sol - contracts/NativeVault.sol - contracts/StBB.sol - contracts/StBBTC.sol	

Description

There are privileged owner roles in the SystemConfig, NativeVault, StBB and StBBTC contracts. The owner can:

1. setLSD, setStakingPool, setNativeVault, setWithdrawalVault, setRewardsVault and setUnstakePeriod in the SystemConfig contract.
2. Toggle isPaused to prevent the contract from being used in the StBB and StBBTC contract.
3. Withdraw all ERC20 tokens and native tokens to the manager in the NativeVault contract.

If the owner's private key is compromised, an attacker could exploit the owner's privileged functions to disrupt the project, which could damage the team's reputation.

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 acknowledged by the team.

2.3 Informational Findings

2. Mismatch between code and comment	
Severity: Informational	Category: Code Quality
Target: - contracts/precompiles/stateless/Bech32.sol - contracts/DelegatorFactory.sol	

Description

1.
contracts/precompiles/stateless/Bech32.sol:L5

```
address constant Bech32_PRECOMPILE_ADDRESS = 0x0000000000000000000000000000000000000400;
```

The Bech32_PRECOMPILE_ADDRESS is set to address(0x0400).

contracts/precompiles/stateless/Bech32.sol:L11

```
/// @custom:address 0x0000000000000000000000000000000000000010
interface Bech32I {
```

However, the comment above indicates that the Bech32 precompile is at address(0x10).

2.
contracts/DelegatorFactory.sol:L98

```
function collectAllFund() external override onlyManagerOrLSD {
```

Both Manager and LSD can call the collectAllFund() function.

contracts/DelegatorFactory.sol:L97

```
/// @dev Only the manager can call this function.
```

However, the comment above indicates that only the manager can call this function.

Recommendation

Consider fixing the mismatch between code and comment.

Status

This issue has been acknowledged by the team.

3. Could use a two-step ownership transfer	
Severity: Informational	Category: Business logic
Target: - contracts/Delegator.sol	

Description

The Delegator contract uses the OwnableUpgradeable contract for ownership management.

It is recommended to use Ownable2StepUpgradeable for transferring ownership. Ownable2StepUpgradeable ensures that the recipient confirms ownership, preventing the transfer of permissions to an incorrect or non-existent address.

Recommendation

Consider using the Ownable2StepUpgradeable provided by openzeppelin.

Status

This issue has been acknowledged by the team.

4. Missing zero address checks	
Severity: Informational	Category: Data Validation
Target: - contracts/NativeVault.sol - contracts/RewardsVault.sol - contracts/StBB.sol - contracts/StBBtc.sol - contracts/WithdrawalVault.sol - contracts/NativeStakingPool.sol - contracts/StakingRewards.sol - contracts/TokenStakingPool.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.

contracts/NativeVault.sol:L32, contracts/RewardsVault.sol:L19, contracts/StBB.sol:L28, contracts/StBBtc.sol:L37, contracts/WithdrawalVault.sol:L31

```
systemConfig = ISystemConfig(_systemConfig);
```

contracts/NativeStakingPool.sol:L15

```
_initialize(_systemConfig, _stakingContract, _distributionContract, _bech32Contract);
```

contracts/StakingRewards.sol:L65

```
stakingToken = IERC20(_stakingToken);
```

contracts/TokenStakingPool.sol:L30-L33

```
_initialize(_systemConfig, _stakingContract, _distributionContract, _bech32Contract);
lsdToken = _lsdToken;
nativePriceFeed = IPriceFeed(_nativePriceFeed);
tokenPriceFeed = IPriceFeed(_tokenPriceFeed);
```

Recommendation

Consider adding zero address checks for address variables.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 068bb17:

File	SHA-1 hash
contracts/Delegator.sol	095a6002836c4f24f2f130d3c8d2e51d5f75baac
contracts/DelegatorFactory.sol	17a452781c395bfd3e45621e2b7cfa032640779b
contracts/NativeStakingPool.sol.sol	2f7bd94fbd5127f5083da33231ae198cc4d0ef60
contracts/NativeVault.sol	b5259910328600cbc1f22faaa2b6120c3f02ea1a
contracts/RewardsVault.sol	3a6d2f49d828f0cca65b2f36cdd647f4cfff28d1
contracts/StBB.sol	14ca57784238a80d76445ddf7f4dffc1941e4a76
contracts/StBBTC.sol	372d3bfb76933b17aef3ab3d3b7e272fe91380ed
contracts/StToken.sol	2d928d828e7033f413df326b82204ec840047058
contracts/StakingPool.sol	7d3da1c2c4e52103abb1e98c41066e6c1b9a9908
contracts/StakingRewards.sol	84b2c756bd121378b20e41d5b12db5726b825556
contracts/SystemConfig.sol	0e2edf1ed8d70e6fb1b58c3afab59d51467bc790
contracts/TokenStakingPool.sol	ace95cfd20320ec5ca9f68339b344614ec7e7801
contracts/WithdrawalQueue.sol	646263e797c8f7617cc34268765750a93d55893f
contracts/WithdrawalVault.sol	c50d14f5aad79367fc971ef78ea17770108f6a85
contracts/precompiles/common/Authorization.sol	d19e86dc875afad13b1744a80b20dd9b47de6836
contracts/precompiles/common/Types.sol	ec57ab9cb6b235e1e97270605fe97c9399802505
contracts/precompiles/stateful/Distribution.sol	53db0e7cf0a14f3a12a30c9b348a5614b9d7f8dd
contracts/precompiles/stateful/Staking.sol	b01524b89e1fe34a42e897f2c8a1e8aef20f59f1
contracts/precompiles/stateless/Bech32.sol	758505b97bad288e84101e7b5bbd5af0c8f12339