SALUS SECURITY

NOV 2024

CODE SECURITY ASSESSMENT

INCEPTIONLRT

Overview

Project Summary

- Name: InceptionLRT
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
  - https://github.com/inceptionlrt/smart-contracts
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	InceptionLRT
Version	v2
Type	Solidity
Dates	Nov 07 2024
Logs	Nov 04 2024; Nov 07 2024

Vulnerability Summary

Total High-Severity issues	1
Total Medium-Severity issues	0
Total Low-Severity issues	3
Total informational issues	4
Total	8

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	Lack of authority verification	High	Business Logic	Resolved
2	Centralization risk	Low	Centralization	Acknowledged
3	Referral mechanism error	Low	Business logic	Resolved
4	Redundancy functions	Informational	Redundancy	Resolved
5	Missing zero address checks	Informational	Data Validation	Resolved
6	Missing two-step transfer ownership pattern	Informational	Business logic	Resolved

2.2 Notable Findings

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

1. Lack of authority verification

Severity: High	Category: Access Control
Target: - projects/restaking-pool/contracts/NativeRebalancer.sol	

Description

projects/restaking-pool/contracts/NativeRebalancer.sol:L254 - L283

```
function handleL2Info(
    uint256 _chainId,
    uint256 _timestamp,
    uint256 _balance,
    uint256 _totalSupply
) external {
    require(
        _timestamp <= block.timestamp,
        TimeCannotBeInFuture(_timestamp)
    );

    Transaction memory lastUpdate = txs[_chainId];

    if (lastUpdate.timestamp != 0) {
        require(
            _timestamp > lastUpdate.timestamp,
            TimeBeforePrevRecord(_timestamp)
        );
    }

    Transaction memory newUpdate = Transaction({
        timestamp: _timestamp,
        ethBalance: _balance,
        inceptionTokenBalance: _totalSupply
    });

    txs[_chainId] = newUpdate;

    emit L2InfoReceived(_chainId, _timestamp, _balance, _totalSupply);
}
```

The function `handleL2Info` is used to update the transaction data of a specific chain and should have been called by `CrossChainAdapter`, but this function does not have any permission control, resulting in the fact that anyone can modify the data of the corresponding chain at will.This issue causes the ` updateTreasuryData` function to make serious errors when balancing the number of L1 and L2 tokens.

Recommendation

It is recommended that strict privilege control be applied to this function.

Status

The team has resolved this issue in commit f313fba.

2. Centralization risk

Severity: Low	Category: Centralization
Target: - projects/restaking-pool/contracts/NativeRebalancer.sol	

Description

There is an `owner` privileged account in the `NativeRebalancer` contract, and the `owner` can change the value of the `lockboxAddress` variable in the contract at will. If `owner`'s private key is compromised, an attacker can change `lockboxAddress` to his own address and subsequently call the `updateTreasuryData` function to obtain an equivalent amount of tokens on all L2 chains.

This could be worrisome if the privileged account is a regular EOA account.

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.

3. Referral mechanism error

Severity: Low	Category: Business Logic
Target: - projects/vaults/contracts/vaults/InceptionOmniVault.sol	

Description

projects/vaults/contracts/vaults/InceptionOmniVault.sol:L142 - L148

```
function depositWithReferral(
    address receiver,
    bytes32 code
) external payable nonReentrant whenNotPaused returns (uint256) {
    emit ReferralCode(code);
    return _deposit(msg.value, msg.sender, receiver);
}
```

The `depositWithReferral` function in the contract is used to perform the `_deposit` operation that contains the referral mechanism, but the `ReferralCode` event only contains the `code` parameter passed in by the user, which leads to reading the event down the chain without being able to determine who triggered the event, allowing the user to refer themselves to get the referral reward.

Recommendation

It is recommended to add `msg.sender` to the event.

Status

The team has resolved this issue in commit 9ead766.

2.3 Informational Findings

4. Redundancy functions	
Severity: Informational	Category: Redundancy
Target: - projects/vaults/contracts/vaults/InceptionOmniVault.sol	

Description

projects/vaults/contracts/vaults/InceptionOmniVault.sol: L361 - L363

```
function getFlashCapacity() public view returns (uint256 total) {
    return totalAssets() - depositBonusAmount;
}
```

projects/vaults/contracts/vaults/InceptionOmniVault.sol: L369 - L371

```
function getTotalDeposited() public view returns (uint256) {
    return totalAssets() - depositBonusAmount;
}
```

Define functions with different meanings: `` `getFlashCapacity` `` and `` `getTotalDeposited` ``, but with the same implementation.

Recommendation

Consider modifying the functionality of one of the functions.

Status

The team has resolved this issue in commit 9ead766.

5. Missing zero address checks

Severity: Informational	Category: Data Validation
Target: - projects/vaults/contracts/vaults/InceptionOmniVault.sol	

Description

projects/vaults/contracts/vaults/InceptionOmniVault.sol: L474 - L477

```
function setOperator(address _newOperator) external onlyOwner {
    emit OperatorChanged(operator, _newOperator);
    operator = _newOperator;
}
```

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 the address variable `operator`.

Recommendation

Add validation in the function to check whether `_newOperator` is zero.

Status

The team has resolved this issue in commit 9ead766.

6. Missing two-step transfer ownership pattern

Severity: Informational	Category: Business logic

Target:
- projects/bridge-lz/contracts/LZCrossChainAdapterL1.sol
- projects/bridge-lz/contracts/LZCrossChainAdapterL2.sol
- projects/restaking-pool/contracts/NativeRebalancer.sol
- projects/vaults/contracts\assets-handler/InceptionAssetsHandler.sol

Description

The `LZCrossChainAdapterL1`, ` LZCrossChainAdapterL2`, `NativeRebalancer`, `InceptionAssetsHandler` contract inherits from the `OwnableUpgradeable` contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

Recommendation

Consider using the Ownable2StepUpgradeable contract from OpenZeppelin instead.

Status

The team has resolved this issue in commit bdc8dd0.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit d22f034:

File	SHA-1 hash
projects/bridge-lz/contracts/abstract/AbstractCrossChainAdapter.sol	2fd709816cd57dce23b70ce80376fea7a8711da5
projects/bridge-lz/contracts/abstract/AbstractCrossChainAdapterL1.sol	73ce656593382099a3ed351f8e3587c7c8122342
projects/bridge-lz/contracts/abstract/AbstractCrossChainAdapterL2.sol	962947f29e5026ef0fa38a647371651b68e74b82
projects/bridge-lz/contracts/abstract/AbstractLZCrossChainAdapter.sol	7eb0359d837a298d20f23beda4ced87a1c78be96
projects/bridge-lz/contracts/LZCrossChainAdapterL1.sol	cbd31eb18f476c606abe0fe052bb0516eb9af4c7
projects/bridge-lz/contracts/LZCrossChainAdapterL2.sol	94858f8cebb3ba433cb9ecd6786b6e0e29312475
projects/vaults/contracts/vaults/InceptionOmniVault.sol	5ce54c0cf63ffdcf237a8f1cbf2fbab017df18be
projects/vaults/contracts/assets-handler/InceptionAssetsHandler.sol	750771e530ad6af5943aad2d6907f4cf26ae689e
projects/restaking-pool/contracts/NativeRebalancer.sol	edf81a10be6c2943361e7273c0971c24c0872f39
projects/restaking-pool/contracts/cToken.sol	5f31bfa85a31b8d09e2ab59cec8582053818a6b2