SALUS SECURITY

JUN 2024

CODE SECURITY ASSESSMENT

LORENZO STAKEPLAN

Overview

Project Summary

- Name: Lorenzo - StakePlan
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/Lorenzo-Protocol/Lorenzo_StakePlan
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Lorenzo - StakePlan
Version	v2
Type	Solidity
Dates	Jun 28 2024
Logs	Jun 27 2024; Jun 28 2024

Vulnerability Summary

Total High-Severity issues	5
Total Medium-Severity issues	1
Total Low-Severity issues	3
Total informational issues	1
Total	10

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	Stakers receive less stBTC due to the difference between decimals	High	Business Logic	Resolved
2	Possible renew stake plan dos via leaving a few unclaimed stBTC	High	Denial of Service	Resolved
3	Dos of renewing stake plan via token donation	High	Denial of Service	Resolved
4	Subscription time limitation does not take effect	High	Business Logic	Resolved
5	The openClaim mechanism does not take effect in one renew stake plan	High	Business Logic	Resolved
6	Centralization risk	Medium	Centralization	Acknowledged
7	Reorg attack to steal funds from bridge	Low	Business Logic	Acknowledged
8	Missing events for functions that change critical states	Low	Business Logic	Resolved
9	Suggest two step permission transfer	Low	Business Logic	Acknowledged
10	Use of floating pragma	Informational	Compiler	Resolved

2.2 Notable Findings

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

1. Stakers receive less stBtc tokens due to the difference in decimals

Severity: High	Category: Business Logic
Target: - StakePlan/StakePlanHub.sol	

Description

The logic of the project assumes that stakers will stake WBTC/BTCB tokens and withdraw stBTC with the same amount. The problem is that the WBTC token decimal is 8 in some chains, while the stBTC token has 18 decimals. The stakers will take high losses.

In this case, when a user staking 1 WBTC, he receives only 0.0000000001 stBTC.

contracts/StakePlan/StakePlanHub.sol:L438-L472

```
function stakeBTC2JoinStakePlan(
   uint256 planId_,
   address btcContractAddress_,
   uint256 stakeAmount
) external override whenNotPaused {
   ...
   IERC20(btcContractAddress_).safeTransferFrom(
       msg.sender,
       derivedStakePlanAddr,
       stakeAmount
   );
   // Mint stakeAmount stBTC ERC20 for stake plan contract 1:1
   IstBTCMintAuthority(_stBTCMintAuthorityAddress).mint(
       derivedStakePlanAddr,
       stakeAmount
   );
   // accounting in stakePlan
   IStakePlan(derivedStakePlanAddr).recordStakeStBTC(
       msg.sender,
       stakeAmount
   );
}
```

Recommendation

It is necessary to take into account the decimals of the token that the user stakes.

Status

This issue has been resolved by the team with commit f8d59ec.

2. Possible renew stake plan dos via leaving a few unclaimed stBTC

Severity: High	Category: Denial of Service
Target: - contracts/StakePlan/StakePlan.sol	

Description

When the plan is renewed, a check is made that totalRaisedStBTC should equal 0, otherwise the renewal will not take place. Malicious users can choose the amount when they want to claim stBTC and leave 1 wei into the contract. Then the function `reNewStakePlan()` will be reverted.

contracts/StakePlan/StakePlan.sol:L104-L121

```
function reNewStakePlan(
        DataTypes.CreateNewPlanData calldata vars_
    ) external onlyHub {
    if (vars_.subscriptionStartTime < _endTime) {
        revert CanNotRenewBeforeEndTime();
    }
    if (totalRaisedStBTC > 0) {
        revert CanNotRenewBeforeStBTCClaimed();
    }
    ......
}
```

Recommendation

If the admin wants to renew one stake plan and some stakers don't claim their stBTCs, we can let stakers claim stBTCs when the next stake plan finishes, or add some states to record unclaimed stBTC and allow them to claim even if we renew one stake plan.

Status

This issue has been resolved by the team with commit f8d59ec.

3. DOS of renewing stake plan via token donation

Severity: High	Category: Denial of Service
Target: - StakePlan/StakePlanHub.sol	

Description

When the admin wants to renew the stake plan, we will check each `_btcContractAddressSet`'s balance in the stake plan should be 0. This check is too strict and easy to exploit.

An attacker can donate one wei WBTC or BTCB into the stake plan contract via frontrun `reNewStakePlan()`. This will block the reNewStakePlan() function.

contracts/StakePlan/StakePlan.sol:L342-L381

```
function reNewStakePlan(
    uint256 planId_,
    DataTypes.CreateNewPlanData calldata vars_
) external override onlyLorenzoAdmin {
    address derivedStakePlanAddr = _stakePlanMap[planId_];
    if (derivedStakePlanAddr == address(0)) {
        revert InvalidPlanId();
    }
    // Parameter check
    if (
        block.timestamp >= vars_.subscriptionStartTime ||
        vars_.subscriptionStartTime >= vars_.subscriptionEndTime ||
        vars_.subscriptionEndTime >= vars_.endTime
    ) {
        revert InvalidSubscriptionTime();
    }
    for (uint256 i = 0; i < _btcContractAddressSet.length(); i++) {
        uint256 balance = IERC20(_btcContractAddressSet.at(i)).balanceOf(
            derivedStakePlanAddr
        );
        if (balance > 0) {
            revert CanNotRenewIfBTCBalanceNotZero();
        }
    }
    IStakePlan(derivedStakePlanAddr).reNewStakePlan(vars_);
}
```

Recommendation

Use internal accounting instead of balanceOf().

Status

This issue has been resolved by the team with commit f8d59ec.

4. Subscription time limitation does not take effect

Severity: High	Category: Business Logic
Target: - StakePlan/StakePlanHub.sol	

Description

When the admin wants to create one new plan, we will assign the stake plan's start time `subscriptionStartTime` and end time `subscriptionEndTime`.

The vulnerability is that the stake plan's start time and end time don't take effect when stakers want to stake tokens. They can stake tokens before start time or after the end time.

contracts/StakePlan/StakePlan.sol:L397-L408

```
function createNewPlan(DataTypes.CreateNewPlanData calldata vars_) external override
whenNotPaused onlyLorenzoAdmin returns (uint256) {
    if (
        block.timestamp >= vars_.subscriptionStartTime ||
        vars_.subscriptionStartTime >= vars_.subscriptionEndTime ||
        vars_.subscriptionEndTime >= vars_.endTime
    ) {
        revert InvalidSubscriptionTime();
    }
    return _createNewPlan(vars_);
}
```

contracts/StakePlan/StakePlan.sol:L438-L472

```
function stakeBTC2JoinStakePlan(
    uint256 planId_,
    address btcContractAddress_,
    uint256 stakeAmount
) external override whenNotPaused {
    ......
    IERC20(btcContractAddress_).safeTransferFrom(
        msg.sender,
        derivedStakePlanAddr,
        stakeAmount
    );
    // Mint stakeAmount stBTC ERC20 for stake plan contract 1:1
    IstBTCMintAuthority(_stBTCMintAuthorityAddress).mint(
        derivedStakePlanAddr,
        stakeAmount
    );
    // accounting in stakePlan
    IStakePlan(derivedStakePlanAddr).recordStakeStBTC(msg.sender,stakeAmount);
}
```

Recommendation

Add start time and end time check in function stakeBTC2JoinStakePlan().

Status

This issue has been resolved by the team with commit 2d32acf.

5. The openClaim mechanism does not take effect in one renew stake plan

Severity: High	Category: Business Logic
Target: - StakePlan/StakePlanHub.sol	

Description

The withdraw process should be like as below:

The gov withdraws WBTC/BTCB tokens and converts to native BTC, bridge to lorenzo chain, mint stBTC in lorenzo bridge contract, then open withdraw for users to claim stBTC.

The vulnerability is that this openClaim mechanism doesn't consider the renewed stake plan case. Once we open the claim in the first round of stake plan, we cannot close the claim. When we renew a second round stake plan, the `canWithdraw` will keep true.

contracts/StakePlan/StakePlan.sol:L417-L424

```
function openClaimStBTC(uint256 planId_) external onlyLorenzoAdmin {
   address derivedStakePlanAddr = _stakePlanMap[planId_];
   if (derivedStakePlanAddr == address(0)) {
       revert InvalidPlanId();
   }
   IStakePlan(derivedStakePlanAddr).openClaimStBTC();
   emit OpenClaimStBTC(planId_);
}
```

contracts/StakePlan/StakePlan.sol:L479-L492

```
function claimStakeStBTC(
   address staker_,
   uint256 amount_
) external onlyHub {
    if (!canWithdraw || block.timestamp < _subscriptionEndTime) {
       revert ClaimstBTCNotOpen();
    }
    userStBTCRecord[staker_] -= amount_;
    totalRaisedStBTC -= amount_;
    IERC20(ST_BTC).safeTransfer(staker_, amount_);
}
```

Recommendation

It is necessary to add a function, which could deny users claim stBTC tokens.

Status

This issue has been resolved by the team with commit 37a9c43.

6. Centralization risk

Severity: Medium	Category: Centralization
Target: - StakePlan/StakePlanHub.sol - stBTC/stBTC.sol - stBTC/stBTCMintAuthority.sol	

Description

The `admin/gov` address holds the ability to negatively impact the system in numerous ways, including but not limited to:

In the StakePlanHub contract, there is a privileged LorenzoAdmin role. The user with this role has the ability to turn on pause mode in contract (this will prohibit stBTC claiming) and modify some key parameters.

In stBTC and stBTCMintAuthority contracts, there is Owner role, The user with this role has the ability to mint any number of tokens.

If the admin's private key is compromised, the attacker can exploit the admin's role to transfer the assets of anyone in the protocol. In this case, the user's assets in the protocol will be seriously threatened.

Recommendation

We recommend transferring the admin role to a multisig account with a timelock feature for enhanced security. This ensures that no single person has full control over the account and that any changes must be authorized by multiple parties.

Status

This issue has been acknowledged by the team.

7. Reorg attack to steal funds from bridge

Severity: Low	Category: Business Logic
Target: - contracts/stBTCBridge/Bridge.sol	

Description

In the Bridge contract, users can bridge their stBTC from one chain to another chain. Users need to burn their stBTC in the source chain, and bridge bots will help to mint related amounts of stBTC in the destination chain.
Checking the project owner, the whole bridging process needs 1 or 2 minutes.
Attach Scenario
Considering below special case:
1. Alice burns stBTC in chain A.
2. Bridge bots mint stBTC in chian B in 1 or 2 minutes.
3. Some unexpected conditions happen, chain A's several or more blocks need to be reorg. Then Alice can transfer her stBTC to another address via frontrun to let step1's transaction fail.
4. Alice can steal money from chain B.
contracts/stBTCBridge/Bridge.sol:L187-L234

```
function burnOrStakeStBtc(
   uint256 amount,
   uint256 toChainId,
   address receiver
) external payable whenNotPaused {
   ......
   if (_supportChain[_chainId].stBTCAddress == NATIVE_TOKEN) {
       if (msg.value != amount + cross_chain_fee) {
           revert InvalidNativeTokenAmount();
       }
   } else {
       if (msg.value != cross_chain_fee) {
           revert InvalidNativeTokenAmount();
       }
       IERC20MintBurnable(stBTCAddress).burnFrom(msg.sender, amount);
   }
   (bool success, ) = payable(_protocolFeeAddress).call{
       value: cross_chain_fee
   }("");
   if (!success) {  revert SendETHFailed();  }
}
```

Recommendation

Different chains may have different characteristics. We need to think about the bridge completion time carefully according to the different chains.

Status

This issue has been acknowledged by the team. The team has stated that the off-chain program will wait for enough blocks to ensure that the blockchain does not reorg

8. Missing events for functions that change critical states

Severity: Low	Category: Logging
Target: - StakePlan/StakePlanHub.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 functions `addSupportBtcContractAddress()` and `removeSupportBtcContractAddress()` events are lacking in the privileged setter functions.

contracts/StakePlan/StakePlan.sol:L252-L262

```
function addSupportBtcContractAddress(
    address[] memory btcContractAddress_
) external onlyGov {
    for (uint256 i = 0; i < btcContractAddress_.length; i++) {
        address btcContractAddress = btcContractAddress_[i];
        if (btcContractAddress == address(0)) {
            revert InvalidAddress();
        }
        _btcContractAddressSet.add(btcContractAddress);
    }
}
```

contracts/StakePlan/StakePlan.sol:L270-L280

```
function removeSupportBtcContractAddress(
    address[] memory btcContractAddress_
) external onlyGov {
    for (uint256 i = 0; i < btcContractAddress_.length; i++) {
        address btcContractAddress = btcContractAddress_[i];
        if (btcContractAddress == address(0)) {
            revert InvalidAddress();
        }
        _btcContractAddressSet.remove(btcContractAddress);
    }
}
```

Recommendation

It is recommended to emit events for critical state changes.

Status

This issue has been resolved by the team with commit f8d59ec.

9. Suggest two step permission transfer

Severity: Low	Category: Business Logic
Target: - StakePlan/StakePlanHub.sol	

Description

The gov address carries numerous important abilities for the system. However the `changeAdmin()` function allows the admin address to be errantly transferred to the wrong address as it does not use a two-step transfer process.

contracts/StakePlan/StakePlanHub.sol:L222-L229

```
function setGovernance(address newGov_) external onlyGov {
    if (newGov_ == address(0)) {
        revert InvalidAddress();
    }
    address preGovernance = _governance;
    _governance = newGov_;
    emit GovernanceSet(preGovernance, _governance);
}
```

Recommendation

Implement a two step “push” and “pull” admin transfer process. If it is desired to have a method to relinquish ownership, implement a separate function to do so.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

10. Use of floating pragma

Severity: Informational	Category: Compiler
Target: - All files	

Description

Lorenzo protocol 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. 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 with commit f8d59ec.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit c4667fc:

File	SHA-1 hash
StakePlanHub.sol	3ad9bff360ca3248a8038a7597f101e924db7ff5
StakePlan.sol	ed4d818243abc6313e4116d47d039e87d7e31f4f
stBTCMintAuthority.sol	c7a93df95c820d98b6ff39f472b0fc0478e1878f
stBTC.sol	2aab3185315573200ba9e745e9a048315e9df10d
BridgeStorage.sol	41bbeddd6c8750d7ff3949fbe449a47d0e86fda7
StakePlanHubStorage.sol	402e4d6c1ed7d39b0c9cb9350ed5388af5bd5c96
Bridge.sol	b3a4bc284d9a6af713a9c059172093d943d1314e
DataTypes.sols	a94bd34d049a7e08963bd32b757f1a184a7d4c10