Lorenzo Protocol

Audit Report

Mon Jul 01 2024

contact@bitslab.xyz

https://twitter.com/scalebit_

ScaleBit

Lorenzo Protocol Audit Report

1 Executive Summary

1.1 Project Information

Description	BTC Liquid staking platform
Type	DeFi
Auditors	ScaleBit
Timeline	Wed Jun 19 2024 - Sun Jun 30 2024
Languages	Solidity
Platform	Ethereum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Lorenzo-Protocol/Lorenzo_StakePlan
Commits	516e2996cb6dd3d03f26abf3761e91d55104ee58 1a6dbbe19b0d655733fc1162804c521ec1e9ea2b 9c296bee9e0c37bbe825c4c0aaed0d57383332a7

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
DTY	contracts/libraries/DataTypes.sol	a94bd34d049a7e08963bd32b757f1a184a7d4c10
BRI	contracts/stBTCBridge/Bridge.sol	df5fff878cbbb047aa77028d1c55339118e75517
SBTCMA	contracts/stBTC/stBTCMintAuthority.sol	e260830f14aefaa6c66a47340402f4aeec947ebb
SBTC	contracts/stBTC/stBTC.sol	788898e892a63ff279e3806682b46ab22d19f445
ISPH	contracts/interfaces/IStakePlanHub.sol	f67699c0bdcea35827ca3d14840f6250f6c22241
IERC2MB	contracts/interfaces/IERC20MintBurnable.sol	74a2d7a2dc889a0074493b083c2bb62642ed3350
ISP	contracts/interfaces/IStakePlan.sol	a679e9df84c3b3309cbc6bc622a5420b72b7a8db
IBTCMA	contracts/interfaces/IstBTCMintAuthority.sol	ba571f9965af7ffea9fdde53bfd66c8523c92692
BST	contracts/storage/BridgeStorage.sol	ef6a9888ffc12b99539a7c0d233dc85b25940d33
SPHS	contracts/storage/StakePlanHubStorage.sol	f9d49c5bc2d4105c9f22403c7b54f456c0160b8a
SPH	contracts/StakePlan/StakePlanHub.sol	08a627defcc5a123543956ac7ea40644996228cd

SPL	contracts/StakePlan/StakePlan.sol	ed4d818243abc6313e4116d47d039e87d7e31f4f

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	13	5	8
Informational	5	0	5
Minor	4	3	1
Medium	3	2	1
Major	1	0	1
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Lorenzo Protocol to identify any potential issues and vulnerabilities in the source code of the Lorenzo Protocol smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 13 issues of varying severity, listed below.

ID	Title	Severity	Status
BRI-1	Lack of Proof of Transaction	Medium	Acknowledged
BRI-2	Missing `receive` Function	Medium	Fixed
BRI-3	`setSupportChainId` Function Setting Risks	Informational	Acknowledged
SBT-1	`revokeRole` Admin	Informational	Acknowledged
SPH-1	Centralization Risk	Major	Acknowledged
SPH-2	Expiration Limit Conflict	Medium	Fixed
SPH-3	Block Time Changes	Minor	Acknowledged
SPH-4	`claimStakeStBTC` Function Missing Checks	Minor	Fixed
SPH-5	Repeat Add	Informational	Acknowledged
SPH-6	The Contract Address Can Be Deleted Before Withdrawal	Informational	Acknowledged
SPH-7	Cannot Set `stBTC` to `_btcContractAddressSet`	Informational	Acknowledged

SPL-1	Lack of Events Emit	Minor	Fixed
SPL-2	Reentrancy Risk	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Lorenzo Protocol Smart Contract :

contracts/StakePlan/StakePlan.sol
Deployer

- Deployer can set the addresses of `STAKE_PLAN_HUB` and `ST_BTC` through the `constructor` function.
- `CreateNewPlanData` will be initialized through the `initialize` function when creating in proxy clone mode.

onlyHub

- `STAKE_PLAN_HUB` can reset `_name` , `_symbol` , `_descUri` , `_agentId` , `_subscriptionStartTime` , `_subscriptionEndTime` , `_endTime` through the `reNewStakePlan` function.
- `STAKE_PLAN_HUB` can record the user's stake amount and the total stake amount within the time limit through the `recordStakeStBTC` function.
- `STAKE_PLAN_HUB` can claim `ST_BTC` to the specified `staker` address through the `claimStakeStBTC` function.
- `STAKE_PLAN_HUB` can record the amount of staker's stake through the `recordStakeStBTC` function.
- `STAKE_PLAN_HUB` can withdraw the balance of `btcContractAddress` in the contract to the `withdrawer` address through the `withdrawBTC` function.
- `STAKE_PLAN_HUB` can open the claiming state through the `openClaimStBTC` function, and it cannot be closed after opening.

contracts/StakePlan/StakePlanHub.sol
Deployer

- After deployment, you need to call the `initialize` function to initialize `Pausable` to implement the pause function, as well as the `_governance` , `_stakePlanImpl` , `_lorenzoAdmin` and `_stBTCMintAuthorityAddress` addresses.

gov

- `_governance` can set the `_lorenzoAdmin` address through the `setLorenzoAdmin` function.

- `_governance` can set `_stBTCMintAuthorityAddress` address through `setStBTCMintAuthorityAddress` function.

- `_governance` can set `_governance` address through `setGovernance` function.

- `_governance` can add `_btcContractAddress` address through `addSupportBtcContractAddress` function.

- `_governance` can remove `_btcContractAddress` address through `removeSupportBtcContractAddress` function.

- `_governance` can withdraw the balance of `btcContractAddress` in `derivedStakePlanAddr` contract through `withdrawBTC` function.

LorenzoAdmin

- LorenzoAdmin can suspend the contract through `adminPauseBridge` function.

- LorenzoAdmin can resume the contract through `adminUnpauseBridge` function.

- LorenzoAdmin can set a new `StakePlan` through the `reNewStakePlan` function.

- LorenzoAdmin can set the creation of `StakePlan` and the corresponding `derivedStakePlanAddr` contract through the `createNewPlan` function.

- LorenzoAdmin can turn on the claim status switch of `derivedStakePlanAddr` through the `openClaimStBTC` function.

User

- User can stake `btcContractAddress_` (WBTC/BTCB tokens supported in the whitelist) token through the `stakeBTC2JoinStakePlan` function.

- User can claim the `stBTC` obtained after staking through the `claimStakeStBTC` function.

contracts/stBTC/stBTC.sol

Deployer

- Initialize msg.sender as the contract owner when deploying the contract.

Owner

- Owner can set the `_minter_contract` address through the `setNewMinterContract` function.

onlyMinterContract

- `_minter_contract` can mint stBTC token through the `mint` function.

contracts/stBTC/stBTCMintAuthority.sol
Deployer

- Initialize and specify `admin` as the contract `DEFAULT_ADMIN_ROLE` when deploying the contract, and set `_stBTCAddress`.

DEFAULT_ADMIN_ROLE

- DEFAULT_ADMIN can add `MINTER_ROLE` by setting the `minter` address through the `setMinter` function.
- DEFAULT_ADMIN can remove `MINTER_ROLE` by setting the `minter` address through the `removeMinter` function.
- DEFAULT_ADMIN can add `ROLE` by setting the `minter` address through the `grantRole` function.
- DEFAULT_ADMIN can remove `ROLE` by setting the `minter` address through the `revokeRole` function.

MINTER_ROLE

- Minter can mint stBTC token to the specified `receipt` address through the `mint` function.

contracts/Bridge/Bridge.sol
Deployer

- When the contract is deployed, call the `initialize` function to initialize the specified `owner` parameter to the contract `Owner` address, and set the addresses of `_relayerOrDao`, `protocolFeeAddress`, `stBTCMintAuthorityAddress`, and initialize the pause function and reentry modifier, as well as `_chainId`.

User

- **User** can burn `stBTCAddress` (stBTC token) or lock the platform token to initiate a cross-chain event through the `burnOrStakeStBtc` function.

onlyAuthRelayerOrDaoContract

- `onlyAuthRelayerOrDaoContract` can send platform tokens to the to address or mint `_stBTCMintAuthorityAddress`(stBTC) through the `mintOrUnstakeStBtc` function

Owner

- The Owner can set the `cross_chain_fee` and `stBTCAddress` of the chain through the `setSupportChainId` function.
- The Owner can pause the contract through the `adminPauseBridge` function.
- The Owner can resume the contract through the `adminUnpauseBridge` function.
- The Owner can set the `_relayerOrDao` address through the `changeRelayer` function.
- The Owner can set the `_protocolFeeAddress` address through the `changeProtocolFeeAddress` function.

4 Findings

BRI-1 Lack of Proof of Transaction

Severity: Medium

Status: Acknowledged

Code Location:

contracts/stBTCBridge/Bridge.sol#120

Descriptions:

1. In the `mintOrUnstakeStBtc` function, the `onlyAuthRelayerOrDaoContract` permission is called. The function does not prove the source of the transaction. In cross-chain transactions, if the dao contract address reads the event record, the transaction proof in the `_usedTxid[txHash]` variable may be preempted, and it cannot prove the source of the user's stake in the contract.
2. On the other hand, if it is a non-existent `txHash`, the default value is false, he can still check and mint amount of stBTC, which will have certain risks.

```
    if (_usedTxid[txHash]) {
        revert TxHashAlreadyMint();
    }

    _usedTxid[txHash] = true;
```

Suggestion:

It is recommended to take mitigation measures.

Resolution:

The client replied that the current stage is centralized and relies on the relayer account being trustworthy (onlyAuthRelayerOrDaoContract), and there is no need to provide proof of cross-chain transactions on other chains.

BRI-2 Missing `receive` Function

Severity: Medium

Status: Fixed

Code Location:

contracts/stBTCBridge/Bridge.sol

Descriptions:

The Bridge contract lacks a receive function, and all the platform token in the contract rely on the `burnOrStakeStBtc payable` function. When liquidity is insufficient, users may not be able to withdraw funds, and the project party may not be able to transfer liquidity to the contract.

Suggestion:

It is recommended to confirm the business logic.

Resolution:

Added `receive` function. When liquidity is insufficient, `RelayerOrDao` can be added manually.

BRI-3 `setSupportChainId` Function Setting Risks

Severity: Informational

Status: Acknowledged

Code Location:

contracts/stBTCBridge/Bridge.sol#199

Descriptions:

When the Bridge contract is deployed in multiple chains, it is necessary to set the corresponding chain and the corresponding contract address information through `setSupportChainId`. When setting, the other chain also needs to add the same settings for adaptation. If the address is set incorrectly, the contract may run incorrectly.

Suggestion:

It is recommended to take mitigation measures.

Resolution:

Multi-chain contracts require synchronous cross-setting, and the client has confirmed the setting risks.

SBT-1 `revokeRole` Admin

Severity: Informational

Status: Acknowledged

Code Location:

contracts/stBTC/stBTCMintAuthority.sol

Descriptions:

If there are multiple `DEFAULT_ADMIN_ROLE` in the stBTCMintAuthority contract, multiple administrators can delete other administrators' `DEFAULT_ADMIN_ROLE` through the `revokeRole` function.

Suggestion:

It is recommended to take mitigation measures.

Resolution:

The client ensures that there will only be one default administrator with the `DEFAULT_ADMIN_ROLE`.

SPH-1 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/StakePlan/StakePlanHub.sol#190;

contracts/stBTC/stBTC.sol#40;

contracts/stBTC/stBTCMintAuthority.sol#32

Descriptions:

In the current system, administrators have the following primary operational privileges:

1. In the stBTC contract, the owner can set the minter address, and the set minter address can `mint` `stBTC` at will.

```
function mint(address receipt, uint256 amount) external onlyMinterContract {
    _mint(receipt, amount);
}
```

2. In the stBTCMintAuthority function, `admin` can set minter through the `setMinter` function, and the address with `MINTER_ROLE` can use the `mint` function to mint `stBTC` at will.

3. The `onlyGov` permission address can modify the implementation address of the `StakPlan` contract. The implementation modified in the future may have certain risks because the updated address may contain new logical functions that are not within the audit scope.

4. `Bridge Owner` can arbitrarily set `stBTCAddress` in the cross-chain bridge, which may involve some risks of external address calls.

Suggestion:

It is recommended to use the multi-sig wallets to mitigate the centralized risk.

SPH-2 Expiration Limit Conflict

Severity: Medium

Status: Fixed

Code Location:

contracts/StakePlan/StakePlanHub.sol#371;

contracts/StakePlan/StakePlan.sol#409

Descriptions:

1. The `stakeBTC2JoinStakePlan` function in the StakePlanHub contract has no association with the staking expiration time. When the open state is turned on, users can continue to stake after the `_subscriptionEndTime` corresponding to the planID, and `withdrawBTC` will withdraw the funds staked after the endtime. Users can stake and withdraw funds in the same block. After calling `stakeBTC2JoinStakePlan`, call `claimStakeStBTC` immediately, and users between `starttime-_subscriptionEndTime` need to lock their positions.

```
if (block.timestamp < _subscriptionEndTime)
```

2. In the `createNewPlan` and `reNewStakePlan` functions, the current time must be less than `subscriptionStartTime`, which means that users can participate in staking between `block.timestamp` and `subscriptionStartTime`, and the settlement time is based on `subscriptionEndTime` instead of `endTime`.

Suggestion:

It recommends confirming the business logic and resolving this conflict.

Resolution:

The stake time is limited.

```
block.timestamp > _subscriptionEndTime||block.timestamp < _subscriptionStartTime
```

SPH-3 Block Time Changes

Severity: Minor

Status: Acknowledged

Code Location:

contracts/StakePlan/StakePlanHub.sol#312

Descriptions:

The `reNewStakePlan` function prevents the modification of the function time by checking the time when the balance limit can be modified, and there may be malicious funds transferred into 1 token.

```
  function reNewStakePlan(
...
      for (uint256 i = 0; i < _btcContractAddressSet.length(); i++) {
          uint256 balance = IERC20(_btcContractAddressSet.at(i)).balanceOf(
              derivedStakePlanAddr
          );
          if (balance > 0) {
              revert CanNotRenewIfBTCBalanceNotZero();
          }
      }
```

Suggestion:

It is recommended to take mitigation measures.

Resolution:

The client confirms that there will be no risk to the client.

SPH-4 `claimStakeStBTC` Function Missing Checks

Severity: Minor

Status: Fixed

Code Location:

contracts/StakePlan/StakePlanHub.sol#409

Descriptions:

There is a lack of stakeAmount check in the `claimStakeStBTC` function, and it will not check whether it exceeds the user's `userStBTCRecord[staker_]`. Executing `userStBTCRecord[staker_] -= amount_` will cause an overflow panic.

Suggestion:

It is recommended to check `stakeAmount>0` and check `userStBTCRecord[staker_] -= amount_`.

Resolution:

Added check for `amount`.

```
    if (
        amount_ == 0 ||
        amount_ > userStBTCRecord[staker_] ||
        amount_ > totalRaisedStBTC
    ) {
        revert InitParamsInvalid();
    }
```

SPH-5 Repeat Add

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakePlan/StakePlanHub.sol#205

Descriptions:

1. The `addSupportBtcContractAddress` function can add the same address when adding btcContractAddress, such as [A,A,A,].

```
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);
    }
}
```

2. For set type functions such as `setLorenzoAdmin`, you can set the same address as the old address.

```
if (newLorenzoAdmin_ == address(0)) {
    revert InvalidAddress();
}
```

Suggestion:

It is recommended to take mitigation measures.

SPH-6 The Contract Address Can Be Deleted Before Withdrawal

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakePlan/StakePlanHub.sol#243

Descriptions:

The `withdrawBTC` function relies on `_btcContractAddressSet` as the token address to query the balance when called. If onlygov deletes the address through the `removeSupportBtcContractAddress` function before the balance is withdrawn, the `withdrawBTC` call will fail.

Suggestion:

It is recommended to check the balance before deleting an address.

SPH-7 Cannot Set `stBTC` to `_btcContractAddressSet`

Severity: Informational

Status: Acknowledged

Code Location:

contracts/StakePlan/StakePlanHub.sol#205

Descriptions:

The `addSupportBtcContractAddress` function can set the `stBTC` token address to `btcContractAddress_`, and obtain stBTC as the stake token in the `stakeBTC2JoinStakePlan` function. If the token is set to `stBTC`, the contract will mint stBTC tokens, which will cause infinite Mint and withdrawBTC errors in the execution process.

Suggestion:

It's recommended that you cannot set `stBTC` as `btc Contract Address`

SPL-1 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

contracts/StakePlan/StakePlan.sol#103;

contracts/StakePlan/StakePlanHub.sol#243;

contracts/stBTC/stBTC.sol#30;

contracts/stBTC/stBTCMintAuthority.sol#36,40;

contracts/StakePlan/StakePlanHub.sol#435

Descriptions:

1. The contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues such as `reNewStakePlan` , `recordStakeStBTC` , `withdrawBTC` , `openClaimStBTC` , `setNewMinterContract` , `setMinter` , `removeMinter` , `addSupportBtcContractAddress` , `removeSupportBtcContractAddress` .
2. `derivedStakePlanAddr` is not recorded in the `_createNewPlan` function trigger event. Although it can be queried through the `_stakePlanMap` variable, it is still recommended to add it to track the binding information when the address is created.

Suggestion:

It is recommended to emit events for those important functions and add `derivedCollectionAddr` in the event trigger.

Resolution:

The client accepted our suggestion.

SPL-2 Reentrancy Risk

Severity: Minor

Status: Fixed

Code Location:

contracts/StakePlan/StakePlan.sol#167,147;

contracts/stBTCBridge/Bridge.sol#128,175,178

Descriptions:

In the `mintOrUnstakeStBtc` , `burnOrStakeStBtc` and 'withdrawBTC' functions, due to the unknown token address, there may be a reentrancy risk if the token is callable during a transfer. Although some functions in the contract address have admin-set token addresses, there may still be a risk of reentrancy when using `safeTransfer` or unchecked to addresses.

```
(bool success, ) = payable(to).call{value: amount}("");
```

Suggestion:

It is recommended to add a no-reentrancy modifier.

Resolution:

Added `nonReentrant` decorator.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.