LEEQUID - Staking

Date	September 2023
Auditors	Rai Yang, Chingiz Mardanov

1 Executive Summary

This report presents the results of our engagement with **LEEQUID** to review their **liquid staking solution**.

The review was conducted over two weeks, from **September 4, 2023** to **September 19th**, by **Rai Yang** and **Chingiz Mardanov**. A total of 20 person-days were spent.

LEEQUID is a protocol targeting the **LUKSO** chain and the corresponding **LYX** token, aiming to to become the goto liquid staking solution of the ecosystem.

Most contracts that were part of this engagement were modified from the StakeWise V2 repository that can be found here. Primary modifications from the StakeWise contracts are reward withdrawal mechanisms that is enabled on the LUKSO chain. An unstake mechanism (validator exiting) as well as unstake/stake order matching component for efficient unstaking are also added to the protocol.

2 Scope

Our review focused on the commit hash `c7348f2c421b0c8d4bfacf746bfbb882e18cce13`. The list of files in scope can be found in the Appendix.

2.1 Objectives

Together with the **LEEQUID** team, we identified the following priorities for our review:

1. Correctness of the implementation, consistent with the intended functionality and without unintended edge cases.
2. Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Best Practices, and the Smart Contract Weakness Classification Registry.

3 System Overview

- **Pool** - Main entrypoint contract through which all of the staking is initiated.
- **StakedLyxToken** - Token that represents the users share of the staked LYX token. This token is non-rebasing and the un-staking process is also initiated via this contract by creating and `UnstakeRequest`. The matching logic between the new deposits and the previously created `UnstakeRequests` also happens in this contract.
- **Rewards** - Contract responsible for the distributing and accumulating the rewards generated by the staking operations.

4 Recommendations

4.1 No Setup for Automated Testing of Key Invariants

Description

The codebase contains a test suite to catch regressions and to test functionality using **manually selected test inputs**. However, it seems like there is no setup for automated testing of key system properties/invariants, for instance, using fuzzing, symbolic execution, or formal verification. Such tools are able to

automatically find test inputs that violate key invariants. For this reason, they are starting to become an important factor when assessing the maturity of a codebase (see, for instance, ToB's code maturity evaluation criteria).

Recommendation

We recommend to identify key system invariants (for instance, using our Scribble specification language), and to set up automated security tools, such as Echidna, Forge, and Diligence Fuzzing, to check those invariants. In general, we recommend to use as many tools as possible. Our blog post on fuzzer benchmarking tries to provide some guidance on how to prioritize tools. However, a tool's effectiveness can vary greatly for different system components.

5 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation. The purpose of this section is to identify specific security properties that were validated by the audit team.

5.1 Actors

The relevant actors are listed below with their respective abilities:

- **User** - User stakes LYX and earns rewards from liquidity staking and validator rewards in sLYX as well as unstakes sLYX
- **Oracle** - Off-chain component that monitors and tracks the state of the beacon chain.
- **Orchestrator** - Off-chain component that collects oracle's votes and publishes the reward data, registers new and updates the state of the activated and exited validators as well as manages un-stake operations through exiting validators.
- **Operator** - account that is allowed to add new validators to the system, currently only one operator, the Leequid team.

- **Admin** - account that is allowed to upgrade the contracts as well as onboard new operators.

5.2 Trust Model

In any system, it's important to identify what trust is expected/required between various actors. For this audit, we established the following trust assumptions:

- Oracles track onchain data correctly and timely
- Orchestrator performs reward publishing, validator registration, validator state updating correctly and timely
- Admin manages operator, oracle and orchestrator and performs contract upgrade correctly

6 Findings

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

6.1 No Protection of Uninitialized Implementation Contracts From Attacker Medium ✓ Fixed

Description

In the contracts implement Openzeppelin's UUPS model, uninitialized implementation contract can be taken over by an attacker with `initialize`

function, it's recommended to invoke the `_disableInitializers` function in the constructor to prevent the implementation contract from being used by the attacker. However all the contracts which implements `OwnablePausableUpgradeable` do not call `_disableInitializers` in the constructors

Examples

contracts/tokens/Rewards.sol:L25

```
contract Rewards is IRewards, OwnablePausableUpgradeable, ReentrancyGuardU
```

contracts/pool/Pool.sol:L20

```
contract Pool is IPool, OwnablePausableUpgradeable, ReentrancyGuardUpgrade
```

contracts/tokens/StakedLyxToken.sol:L46

```
contract StakedLyxToken is OwnablePausableUpgradeable, LSP4DigitalAssetMet
```

etc.

Recommendation

Invoke `_disableInitializers` in the constructors of contracts which implement `OwnablePausableUpgradeable` including following:

```
Pool
PoolValidators
FeeEscrow
Reward
StakeLyxTokem
Oracles
MerkleDistributor
```

6.2 Unsafe Function receiveFees Minor Acknowledged

Description

In the Pool contract, function `receiveFees` is used for compensate a potential penalty/slashing in the protocol by sending LYX back to the pool without minting sLYX, but the side effect is that anyone can send LYX to the pool which could mess up pool balance after all validator exited, in fact it can be replaced by a another function `receiveWithoutActivation` with access control which does the same thing.

Examples

contracts/pool/Pool.sol:L153

```
function receiveFees() external payable override {}
```

contracts/pool/Pool.sol:L132-L134

```
function receiveWithoutActivation() external payable override {
    require(msg.sender == address(stakedLyxToken) || hasRole(DEFAULT_ADMIN
}
```

Recommendation

Remove function `receiveFees`

6.3 Unnecessary Matching in Unstake Process Minor ✓ Fixed

Description

Function `unstakeProcessed` in `StakedLyxToken` contract, when `unstakeAmount > totalPendingUnstake`, all the unstake requests should be able to be processed, thus no need to go through the matching, as a result, extra gas in the matching can be saved.

Examples

contracts/tokens/StakedLyxToken.sol:L388-L411

```
if (unstakeAmount > totalPendingUnstake) {
    pool.receiveWithoutActivation{value: unstakeAmount - totalPendingUnsta
    unstakeAmount = totalPendingUnstake;
}

totalPendingUnstake -= unstakeAmount;
totalUnstaked += unstakeAmount;
uint256 amountToFill = unstakeAmount;

for (uint256 i = unstakeRequestCurrentIndex; i <= unstakeRequestCount; i++
    UnstakeRequest storage request = _unstakeRequests[i];
    if (amountToFill > (request.amount - request.amountFilled)) {
        amountToFill -= (request.amount - request.amountFilled);
        continue;
    } else {
        if (amountToFill == (request.amount - request.amountFilled) && i <
            unstakeRequestCurrentIndex = i + 1;
        } else {
            request.amountFilled += uint128(amountToFill);
            unstakeRequestCurrentIndex = i;
        }
        break;
    }
}
```

Recommendation

Put the matching part (line 393-411) into else branch of `if unstakeAmount > totalPendingUnstake`, change the if branch into following:

```
if (unstakeAmount > totalPendingUnstake) {
            pool.receiveWithoutActivation{value: unstakeAmount - totalPendingUnstake}();
            unstakeAmount = totalPendingUnstake;
            totalPendingUnstake = 0;
            unstakeRequestCurrentIndex = unstakeRequestCount;
            _unstakeRequests[unstakeRequestCount].amountFilled = _unstakeRequests[unstakeRe
 }
```

6.4 Redundant Parameter in the Initialize Function

Description

In the `initialize` function of the Pool contract, the parameter `_withdrawalCredentials` is redundant, as all validator's withdrawal credentials are set to the address of the Reward contract and reward contract's address is already set by the initialization parameter `_rewards` .

Examples

contracts/pool/Pool.sol:L59-L76

```
function initialize(
    address _admin,
    address _stakedLyxToken,
    address _rewards,
    address _validators,
    address _oracles,
    bytes32 _withdrawalCredentials,
    address _validatorRegistration,
    uint256 _minActivatingDeposit,
    uint256 _pendingValidatorsLimit
) public initializer {
    require(_stakedLyxToken != address(0), "Pool: stakedLyxToken address c
    require(_rewards != address(0), "Pool: rewards address cannot be zero"
    require(_admin != address(0), "Pool: admin address cannot be zero");
    require(_oracles != address(0), "Pool: oracles address cannot be zero"
    require(_validatorRegistration != address(0), "Pool: validatorRegistra
    require(_validators != address(0), "Pool: validators address cannot be
    require(_withdrawalCredentials != bytes32(0), "Pool: withdrawalCredent
```

Recommendation

Remove the parameter `_withdrawalCredentials` from intialize function

Appendix 1 - Files in Scope

This audit covered the following files:

File	SHA-1 hash
contracts/AdminUpgradeableProxy.sol	`6325b086018baf4323a49458ed2107c409c0ec2a`
contracts/Oracles.sol	`c2f5b8bd673f6f40e5b6978c90d944123266018d`
contracts/merkles/MerkleDistributor.sol	`eed531d2ca2ce7dcd8328d038e72732569aa6768`
contracts/pool/FeesEscrow.sol	`2b9472c2e5b4062ade13975fb507a40f3b53ebae`

File	SHA-1 hash
contracts/pool/Pool.sol	`44df2cf5e2fcd88e8abe9d523a210dbe1280b691`
contracts/pool/PoolValidators.sol	`38af87367ee8e59e11e16cd59797819e1f2a1b1e`
contracts/tokens/Rewards.sol	`3c1b5a01c23ab161c1d55bf3323a2c594e592eb6`
contracts/tokens/StakedLyxToken.sol	`d411f979be4b9df174742b93be2d33982c3da4f0`

Appendix 2 - Disclosure

Consensys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via Consensys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any third party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any third party by virtue of publishing these Reports.

A.2.1 Purpose of Reports

The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

A.2.2 Links to Other Web Sites from This Web Site

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Consensys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that Consensys and CD are not responsible for the content or operation of such Web sites, and that Consensys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that Consensys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. Consensys and CD assumes no responsibility for the use of third-party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

A.2.3 Timeliness of Content

The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice unless indicated otherwise, by Consensys and CD.