SolidLizard Lending
Audit Report

Tue Apr 30 2024

contact@scalebit.xyz https://twitter.com/scalebit_

ScaleBit

SolidLizard Lending Audit Report

1 Executive Summary

1.1 Project Information

Description	SolidLizard lending is a highly scalable decentralized lending protocol powered by Arbitrum.
Type	Lending
Auditors	ScaleBit
Timeline	Mon Apr 22 2024 - Tue Apr 30 2024
Languages	Solidity
Platform	Arbitrum
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/SolidLizard/lend-Staking https://github.com/SolidLizard/SL-lend-contracts
Commits	https://github.com/SolidLizard/lend-Staking: c33a47380c2710b042b04446762a42e908ed5039 https://github.com/SolidLizard/SL-lend-contracts: 0c4c4fc720e9bc3ca8128d41d74a2b669b7d736c

1.2 Files in Scope

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

ID	File	SHA-1 Hash
ICL	contracts/interfaces/IClaimable.sol	6aaf87beeb34d15f943f348efbab120ff8bc736b
CE2	contracts/CErc20.sol	cc29cfee71b2d80b65e79dd65c6a960dc5d8e096
RDI	contracts/RewardDistributor.sol	2ea3e2fb6bc29c21ab1f8fc9752240f5f90b5fd3
CE2U	contracts/CErc20Upgradable.sol	5a621653a967d9190c15f2a5bcc0fcc3805ff02b
CTI	contracts/CTokenInterfaces.sol	9d0e4c03e1056424ba9db662163d6024d066d2a4
ERE	contracts/ErrorReporter.sol	0fa8ca1cb6b7ac10f15349dc20cc661571c3585e
CIN	contracts/ComptrollerInterface.sol	62917396c9b7179f6a73fdd1e6cc230adbc5d58b
UNI	contracts/Unitroller.sol	720c40892523a3c6f58bdb3461a23963e7b969ad
CST	contracts/ComptrollerStorage.sol	9bd23be2f93eea76d8daf6193a07650ddb88f670
IRM	contracts/InterestRateModel.sol	0ff4dda8c2d430452caf0b62028ba93d55d9de15
JRMV4	contracts/JumpRateModelV4.sol	de7031804f839321e06c24db1ccff6df6865d11c

UPO	contracts/PriceOracle/UniswapPriceOracle.sol	91897e1db2e8e411f6fdcc646dd7ec8226af5ee6
SPO	contracts/PriceOracle/SimplePriceOracle.sol	11780ffc4b8a6b6f49aaa61ef5016885974252f0
CPO	contracts/PriceOracle/ChainlinkPriceOracle.sol	093e8fd0c2ef60f38980ff9034e6ca7abead7dd2
WPO	contracts/PriceOracle/WitnetPriceOracle.sol	0009b161955934821a464a0254357e2493bcec84
CE2I	contracts/CErc20Immutable.sol	66843e7b0d51bef4439a0f54a79d9956702e070c
POR	contracts/PriceOracle.sol	ca58bd9b259ee222a8842cf289fbbde396732888
COM	contracts/Comptroller.sol	10a41beeb807a410b39d83af9ba2c7da4d21d089
ENE	contracts/ExponentialNoError.sol	5b1bfa0f01c6642e2c850f45e47c97ecdc015e03
LIN	contracts/interfaces/LendingInterfaces.sol	e315b583d67c1ec483a0f8dbb7218a0605dc9f41
LGEDI	contracts/interfaces/LGEDepositorInterfaces.sol	cdb94cc6ee829354f98453f020b3950060b1505a
VIN	contracts/interfaces/VelocoreInterfaces.sol	c6d89826d234991c624f99a446f68ea395121d45
RMA	contracts/interfaces/RewardManager.sol	eb6199ca8c17c5dd470545b0e3776fffcc5b5c76
RHV3	contracts/RewardHolderV3.sol	5eec0ff2ef4ea14fddb656a8e60bd09d1f446e3c

RHO	contracts/RewardHolder.sol	99508d5ce0f51955ed3e138c4722213779e061ca
SDI	contracts/StakedDistributor.sol	57feb6d25fd3158a59f818c802e3bcf59cccc8d6
ODI	contracts/OwnedDistributor.sol	8b3b394bdbebfc960660af25b2c851094262287b
RHV2	contracts/RewardHolderV2.sol	1d158b955e8eed7e4b09cb330f5dfa80c07116ae
RHV4	contracts/RewardHolderV4.sol	343589c0d15543c1e0b39b1ec81d0b78f94fef5d
RMA1	contracts/RewardManager.sol	613d7d5796056f87ea303c48501774838ec7c817
RMA2	contracts/ReserveManager.sol	2110921f8c1f7f944ef234a8df56eca4ad035074
DIS	contracts/Distributor.sol	514bce5fdf0f0c03ebc62ce10bd2bb5f7564c9e0
FUN	contracts/funds/Fund.sol	dddfa7e0fceb0a4c30134e8c8dfb90378ba390fd
SSAMF	contracts/funds/ScalesStakingAndMiningFund.sol	1b9d5e71064ac29ad053a617f58352dbfd533043
SDF	contracts/funds/ScalesDevFund.sol	f0d9fdee4b4f6154f124ec246a6d09ff19c86150
STF	contracts/funds/ScalesTreasuryFund.sol	4d08bd6efa94614886f28bb59a5d38c0487a9ecf
TCO	contracts/TimelockController.sol	3b1a0bd54940ca89ecd6e6eac4eed471ef048bb4

CTO	contracts/CToken.sol	6b79c047d7b86615355e3252759c2515461cbffa
SLLTC	contracts/SLLendTimelockController.sol	98121f11cabc7053113935bc2ec25828023af2ee
SCA	contracts/Token/Scales.sol	4defbf11ff4e45b8181b8cdebe712f7061c7bb93
XSC	contracts/Token/XScales.sol	ac4586cee585ce177a1f0cc035e124f9adc224c9
PPO	contracts/PriceOracle/PythPriceOracle.sol	0f9f970c8151c0ddc50c6ec40412fc0cc928cce5
MPO	contracts/PriceOracle/MixedPriceOracle.sol	40434719fba2d96758c7a7dab2fad47db91115bc
MPOWWETH	contracts/PriceOracle/MixedPriceOracleWithWstETH.sol	a6c2ab61f6520511074f692bdc93a0b52b9a956d
MPOWWETH2	contracts/PriceOracle/MixedPriceOracleWithWstETHv2.sol	e7726e59cb0bb8f0ca134a5b09d88eb1acae8764

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	3	1	2
Informational	0	0	0
Minor	2	1	1
Medium	1	0	1
Major	0	0	0
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 SolidLizard to identify any potential issues and vulnerabilities in the source code of the SolidLizard Lending 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 3 issues of varying severity, listed below.

ID	Title	Severity	Status
RHV-1	Incompatible With Deflationary Token	Medium	Acknowledged
RHV-2	Code Optimization	Minor	Fixed
XSC-1	Redundant Checking	Minor	Acknowledged

3 Participant Process

Here are the relevant actors with their respective abilities within the SolidLizard Lending Smart Contract :

Admin

- Admin can create a add reward tokens through `_whitelistToken()`.
- Admin can set reward holder where can be claimed through `_setClaimable()`.
- Admin can set the guy who can claim through `_setRecipient()`.
- Admin can set the `admin`, `reservesManager_` of `LiquidityGenerator` through `_setAdmin()`.

Recipient

- Recipient can claim reward coins through `RewardHolder.claim()`.

User

- User can update the reward shareIndex through `updateShareIndex()`.
- User can get their stake coins through `mint()`.
- User can get their underlying coins through `burn()`.
- User can withdraw their underlying coins through `withdraw()`.
- User can deposit their token to get reward through `deposit()`.
- User can get reward tokens through `claim()`.

4 Findings

RHV-1 Incompatible With Deflationary Token

Severity: Medium

Status: Acknowledged

Code Location:

contracts/RewardHolderV2.sol#59;

contracts/RewardHolderV3.sol#62;

contracts/RewardHolderV4.sol#62

Descriptions:

In the `addRewards` function, due to the unknown address of the `token`, when the token is deflationary, the number of tokens transferred to the contract by the user may not be accurate.

Suggestion:

Since it's not known exactly what type of token this is, it's recommended to confirm whether such a question would conflict with the design philosophy.

Resolution:

The client said that they will be mindful of this. Currently, they have no plans to introduce deflationary rewards to their staking and LP Mining contracts.

RHV-2 Code Optimization

Severity: Minor

Status: Fixed

Code Location:

contracts/RewardHolderV2.sol#60,61;

contracts/RewardHolderV3.sol#63,64;

contracts/RewardHolderV4.sol#63,64

Descriptions:

In the `addRewards` function, the incoming token parameter was originally of type address, and it's a redundant step to convert the token to address again when you use it.

Suggestion:

It is recommended to fix it.

Resolution:

The client will update the contract to incorporate this optimization.

XSC-1 Redundant Checking

Severity: Minor

Status: Acknowledged

Code Location:

contracts/Token/XScales.sol#108

Descriptions:

The `lockedAmount` parameter is of type uint256 and will always be greater than or equal to 0.

Suggestion:

It is recommended to delete the check.

Resolution:

The client acknowledged that this check is redundant. However, for security reasons, the `xScales` contract is not upgradeable, so they will retain this check.

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.