ABDK CONSULTING

SMART CONTRACT AUDIT

Lossless

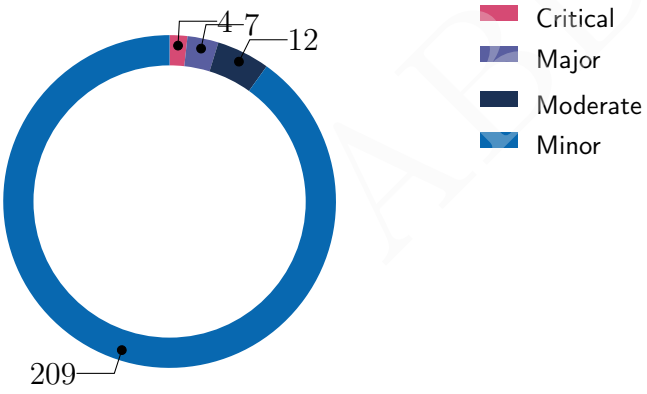
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
22nd February 2022

We've been asked to review the 20 files in a Github repository. We found 4 critical, 7 major, and a few less important issues. All critical issues were fixed. Major issues were fixed or handled during refactoring. The issue CVF-18 was discussed with the client (comment added).



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Fixed
CVF-2	Moderate	Procedural	Fixed
CVF-3	Minor	Bad datatype	Fixed
CVF-4	Minor	Suboptimal	Fixed
CVF-5	Minor	Bad naming	Fixed
CVF-6	Minor	Procedural	Fixed
CVF-7	Minor	Procedural	Fixed
CVF-8	Minor	Bad datatype	Fixed
CVF-9	Minor	Bad datatype	Fixed
CVF-10	Minor	Procedural	Fixed
CVF-11	Minor	Bad datatype	Fixed
CVF-12	Minor	Suboptimal	Fixed
CVF-13	Minor	Suboptimal	Fixed
CVF-14	Minor	Suboptimal	Fixed
CVF-15	Minor	Bad datatype	Fixed
CVF-16	Minor	Readability	Fixed
CVF-17	Moderate	Flaw	Fixed
CVF-18	Major	Unclear Behaviour	Info
CVF-19	Minor	Bad datatype	Fixed
CVF-20	Moderate	Procedural	Fixed
CVF-21	Minor	Readability	Fixed
CVF-22	Minor	Bad datatype	Fixed
CVF-23	Minor	Procedural	Fixed
CVF-24	Minor	Suboptimal	Fixed
CVF-25	Moderate	Procedural	Fixed
CVF-26	Minor	Bad naming	Fixed
CVF-27	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-28	Minor	Procedural	Fixed
CVF-29	Minor	Bad datatype	Fixed
CVF-30	Minor	Bad datatype	Info
CVF-31	Minor	Procedural	Fixed
CVF-32	Minor	Bad datatype	Fixed
CVF-33	Minor	Readability	Info
CVF-34	Minor	Suboptimal	Fixed
CVF-35	Minor	Suboptimal	Fixed
CVF-36	Minor	Suboptimal	Fixed
CVF-37	Minor	Readability	Fixed
CVF-38	Minor	Readability	Fixed
CVF-39	Minor	Suboptimal	Fixed
CVF-40	Minor	Suboptimal	Info
CVF-41	Minor	Suboptimal	Fixed
CVF-42	Minor	Procedural	Fixed
CVF-43	Minor	Readability	Fixed
CVF-44	Minor	Readability	Fixed
CVF-45	Minor	Readability	Info
CVF-46	Minor	Suboptimal	Fixed
CVF-47	Minor	Suboptimal	Fixed
CVF-48	Minor	Suboptimal	Fixed
CVF-49	Minor	Flaw	Fixed
CVF-50	Minor	Suboptimal	Fixed
CVF-51	Minor	Suboptimal	Fixed
CVF-52	Minor	Suboptimal	Fixed
CVF-53	Minor	Suboptimal	Info
CVF-54	Moderate	Flaw	Fixed
CVF-55	Minor	Suboptimal	Info
CVF-56	Minor	Suboptimal	Fixed
CVF-57	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-58	Critical	Flaw	Fixed
CVF-59	Minor	Procedural	Fixed
CVF-60	Moderate	Flaw	Fixed
CVF-61	Minor	Procedural	Fixed
CVF-62	Minor	Suboptimal	Fixed
CVF-63	Minor	Bad naming	Fixed
CVF-64	Minor	Suboptimal	Fixed
CVF-65	Minor	Bad datatype	Fixed
CVF-66	Minor	Suboptimal	Fixed
CVF-67	Minor	Bad datatype	Fixed
CVF-68	Minor	Procedural	Fixed
CVF-69	Minor	Bad datatype	Fixed
CVF-70	Minor	Bad datatype	Fixed
CVF-71	Minor	Procedural	Fixed
CVF-72	Minor	Suboptimal	Fixed
CVF-73	Moderate	Flaw	Fixed
CVF-74	Minor	Bad datatype	Fixed
CVF-75	Minor	Readability	Fixed
CVF-76	Critical	Flaw	Fixed
CVF-77	Minor	Procedural	Fixed
CVF-78	Minor	Procedural	Fixed
CVF-79	Minor	Bad datatype	Fixed
CVF-80	Major	Procedural	Fixed
CVF-81	Minor	Bad datatype	Fixed
CVF-82	Minor	Suboptimal	Fixed
CVF-83	Minor	Suboptimal	Fixed
CVF-84	Minor	Unclear behavior	Info
CVF-85	Minor	Unclear behavior	Info
CVF-86	Minor	Unclear behavior	Fixed
CVF-87	Minor	Suboptimal	Fixed

ID	Severity	Category	Status
CVF-88	Minor	Bad naming	Fixed
CVF-89	Minor	Bad datatype	Fixed
CVF-90	Minor	Procedural	Fixed
CVF-91	Minor	Procedural	Fixed
CVF-92	Minor	Suboptimal	Fixed
CVF-93	Minor	Suboptimal	Fixed
CVF-94	Minor	Bad datatype	Info
CVF-95	Minor	Procedural	Fixed
CVF-96	Minor	Documentation	Fixed
CVF-97	Minor	Suboptimal	Info
CVF-98	Minor	Bad datatype	Fixed
CVF-99	Minor	Bad datatype	Fixed
CVF-100	Minor	Bad datatype	Fixed
CVF-101	Major	Flaw	Fixed
CVF-102	Minor	Suboptimal	Fixed
CVF-103	Minor	Suboptimal	Fixed
CVF-104	Minor	Documentation	Fixed
CVF-105	Major	Unclear behavior	Fixed
CVF-106	Minor	Documentation	Fixed
CVF-107	Minor	Suboptimal	Fixed
CVF-108	Minor	Suboptimal	Fixed
CVF-109	Minor	Readability	Fixed
CVF-110	Minor	Suboptimal	Fixed
CVF-111	Minor	Overflow/Underflow	Fixed
CVF-112	Minor	Suboptimal	Fixed
CVF-113	Minor	Suboptimal	Fixed
CVF-114	Minor	Suboptimal	Fixed
CVF-115	Minor	Readability	Fixed
CVF-116	Minor	Bad datatype	Fixed
CVF-117	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-118	Critical	Flaw	Fixed
CVF-119	Minor	Documentation	Fixed
CVF-120	Minor	Suboptimal	Info
CVF-121	Critical	Flaw	Fixed
CVF-122	Moderate	Unclear behavior	Fixed
CVF-123	Major	Suboptimal	Info
CVF-124	Minor	Suboptimal	Info
CVF-125	Minor	Suboptimal	Info
CVF-126	Minor	Procedural	Fixed
CVF-127	Minor	Procedural	Fixed
CVF-128	Minor	Bad naming	Info
CVF-129	Minor	Bad datatype	Info
CVF-130	Minor	Readability	Info
CVF-131	Minor	Bad datatype	Fixed
CVF-132	Minor	Bad naming	Fixed
CVF-133	Minor	Suboptimal	Info
CVF-134	Minor	Bad datatype	Fixed
CVF-135	Minor	Procedural	Fixed
CVF-136	Moderate	Unclear behavior	Info
CVF-137	Minor	Suboptimal	Fixed
CVF-138	Minor	Suboptimal	Fixed
CVF-139	Minor	Procedural	Info
CVF-140	Minor	Procedural	Fixed
CVF-141	Minor	Procedural	Fixed
CVF-142	Minor	Suboptimal	Fixed
CVF-143	Minor	Procedural	Fixed
CVF-144	Minor	Procedural	Fixed
CVF-145	Minor	Suboptimal	Info
CVF-146	Major	Procedural	Fixed
CVF-147	Minor	Documentation	Fixed

ID	Severity	Category	Status
CVF-148	Minor	Suboptimal	Fixed
CVF-149	Minor	Procedural	Fixed
CVF-150	Minor	Suboptimal	Fixed
CVF-151	Minor	Bad naming	Fixed
CVF-152	Minor	Suboptimal	Info
CVF-153	Minor	Suboptimal	Fixed
CVF-154	Minor	Suboptimal	Fixed
CVF-155	Minor	Suboptimal	Fixed
CVF-156	Minor	Suboptimal	Fixed
CVF-157	Minor	Suboptimal	Fixed
CVF-158	Major	Suboptimal	Fixed
CVF-159	Minor	Readability	Fixed
CVF-160	Minor	Unclear behavior	Info
CVF-161	Minor	Suboptimal	Fixed
CVF-162	Minor	Suboptimal	Fixed
CVF-163	Minor	Suboptimal	Fixed
CVF-164	Minor	Suboptimal	Fixed
CVF-165	Minor	Procedural	Fixed
CVF-166	Minor	Procedural	Info
CVF-167	Minor	Suboptimal	Fixed
CVF-168	Minor	Suboptimal	Fixed
CVF-169	Minor	Bad naming	Fixed
CVF-170	Minor	Suboptimal	Fixed
CVF-171	Minor	Procedural	Info
CVF-172	Minor	Suboptimal	Fixed
CVF-173	Minor	Suboptimal	Fixed
CVF-174	Minor	Procedural	Fixed
CVF-175	Minor	Suboptimal	Info
CVF-176	Minor	Suboptimal	Info
CVF-177	Minor	Procedural	Fixed

ID	Severity	Category	Status
CVF-178	Minor	Bad datatype	Fixed
CVF-179	Minor	Bad datatype	Fixed
CVF-180	Minor	Bad datatype	Fixed
CVF-181	Minor	Bad datatype	Fixed
CVF-182	Minor	Suboptimal	Fixed
CVF-183	Moderate	Flaw	Info
CVF-184	Minor	Bad datatype	Fixed
CVF-185	Minor	Bad datatype	Fixed
CVF-186	Minor	Bad datatype	Fixed
CVF-187	Minor	Bad naming	Fixed
CVF-188	Minor	Bad datatype	Fixed
CVF-189	Minor	Procedural	Fixed
CVF-190	Minor	Procedural	Fixed
CVF-191	Minor	Bad datatype	Fixed
CVF-192	Minor	Bad datatype	Fixed
CVF-193	Minor	Bad datatype	Fixed
CVF-194	Minor	Bad datatype	Fixed
CVF-195	Minor	Bad datatype	Fixed
CVF-196	Minor	Bad datatype	Fixed
CVF-197	Minor	Bad datatype	Fixed
CVF-198	Minor	Bad naming	Fixed
CVF-199	Minor	Procedural	Fixed
CVF-200	Minor	Procedural	Fixed
CVF-201	Moderate	Bad datatype	Fixed
CVF-202	Moderate	Bad datatype	Info
CVF-203	Minor	Bad datatype	Info
CVF-204	Minor	Bad datatype	Fixed
CVF-205	Minor	Bad datatype	Fixed
CVF-206	Minor	Bad datatype	Fixed
CVF-207	Minor	Suboptimal	Info

ID	Severity	Category	Status
CVF-208	Minor	Procedural	Fixed
CVF-209	Minor	Bad naming	Fixed
CVF-210	Minor	Suboptimal	Fixed
CVF-211	Minor	Suboptimal	Fixed
CVF-212	Minor	Suboptimal	Fixed
CVF-213	Minor	Bad datatype	Fixed
CVF-214	Minor	Bad datatype	Fixed
CVF-215	Minor	Procedural	Fixed
CVF-216	Minor	Suboptimal	Info
CVF-217	Minor	Procedural	Fixed
CVF-218	Minor	Bad datatype	Fixed
CVF-219	Minor	Bad naming	Fixed
CVF-220	Minor	Documentation	Fixed
CVF-221	Minor	Documentation	Info
CVF-222	Minor	Bad datatype	Info
CVF-223	Minor	Bad datatype	Fixed
CVF-224	Minor	Bad datatype	Fixed
CVF-225	Minor	Bad datatype	Info
CVF-226	Minor	Bad datatype	Fixed
CVF-227	Minor	Bad datatype	Fixed
CVF-228	Minor	Bad datatype	Fixed
CVF-229	Minor	Suboptimal	Info
CVF-230	Minor	Bad naming	Fixed
CVF-231	Minor	Suboptimal	Fixed
CVF-232	Minor	Procedural	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	February 21, 2022	D. Khovratovich	Initial Draft
0.2	February 21, 2022	D. Khovratovich	Minor revision
1.0	February 21, 2022	D. Khovratovich	Release
1.1	February 22, 2022	D. Khovratovich	CVF-52, typo fixed
2.0	February 22, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the contracts at the repository:

- ILosslessController.sol
- ILosslessERC20.sol
- ILosslessGovernance.sol
- ILosslessReporting.sol
- ILosslessStaking.sol
- IProtectionStrategy.sol
- Context.sol
- LERC20.sol
- LosslessControllerV1.sol
- mockTransfer.sol
- LiquidityProtectionMultipleLimitsStrategy.sol
- LiquidityProtectionSingleLimitStrategy.sol
- LosslessControllerV2.sol
- LosslessGuardian.sol
- StrategyBase.sol
- TreasuryProtectionStrategy.sol
- LosslessControllerV3.sol
- LosslessGovernance.sol
- LosslessReporting.sol
- LosslessStaking.sol

The fixes were provided in the acd9062 commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessReporting.sol

Recommendation Should be "^0.8.0". Also relevant for the next files: LosslessGovernance.sol, LosslessStaking.sol, LosslessControllerV3.sol, LosslessControllerV2.sol, LERC20.sol, LosslessControllerV1.sol, ILosslessStaking.sol, ILosslessGovernance.sol, ILosslessERC20.sol, ILosslessControllerV3.sol, ILosslessReporting.sol.

Listing 1:

```
2 pragma solidity 0.8.0;
```

3.2 CVF-2

- **Severity** Moderate
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessReporting.sol

Recommendation This contract should implement the "ILssReporting" interface to ensure consistency between the contract and the interface.

Listing 2:

```
16 contract LosslessReporting is Initializable, ContextUpgradeable,
    ↪ PausableUpgradeable {
```

3.3 CVF-3

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The key type should be "ILERC20".

Listing 3:

```
35 mapping(address => TokenReports) private tokenReports;
```

3.4 CVF-4

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessReporting.sol

Recommendation It would be more efficient to replace these mappings with a single mapping whose keys are report IDs and values are structs encapsulating the values of the original mappings.

Listing 4:

```
37 mapping(uint256 => bool)  private reporterClaimStatus;

39 mapping(uint256 => address) public reporter;
40 mapping(uint256 => address) public reportedAddress;
   mapping(uint256 => address) public secondReportedAddress;
   mapping(uint256 => uint256) public reportTimestamps;
   mapping(uint256 => address) public reportTokens;
   mapping(uint256 => bool) public secondReports;
```

3.5 CVF-5

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessReporting.sol

Recommendation Events are usually named via nouns such as "ReportSubmission" or "NewReport" or just "Report".

Listing 5:

```
47 event ReportSubmitted(address indexed token, address indexed
     ↪ account, uint256 reportId);
   event SecondReportsubmitted(address indexed token, address
     ↪ indexed account, uint256 reportId);
   event ReportingAmountChanged(uint256 indexed newAmount);
```

3.6 CVF-6

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessReporting.sol

Recommendation The“reportId” parameters should be indexed”.

Listing 6:

```
47 event ReportSubmitted(address indexed token, address indexed
   ↪ account, uint256 reportId);
   event SecondReportsubmitted(address indexed token, address
   ↪ indexed account, uint256 reportId);
```

3.7 CVF-7

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessReporting.sol

Description In the former modifier “msg.sender” is at the right while in the latter it is at the “left”.
Recommendation Consider being more consistent.

Listing 7:

```
55 require(losslessController.admin() == msg.sender, "LSS: Must be
   ↪ admin");

61 require(msg.sender == losslessController.pauseAdmin(), "LSS:
   ↪ Must be pauseAdmin");
```

3.8 CVF-8

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The argument type should be “ILssController”.

Listing 8:

```
78 function initialize(address _losslessController) public
   ↪ initializer {
```

3.9 CVF-9

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The argument type should be "ILERC20".

Listing 9:

```
97 function setStakingToken(address _stakingToken) public
     ↪ onlyLosslessAdmin {
```

3.10 CVF-10

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessReporting.sol

Recommendation These functions should emit some events.

Listing 10:

```
97 function setStakingToken(address _stakingToken) public
     ↪ onlyLosslessAdmin {

105 function setLosslessGovernance(address _losslessGovernance)
     ↪ public onlyLosslessAdmin {

119 function setReporterReward(uint256 reward) public
     ↪ onlyLosslessAdmin {

126 function setLosslessReward(uint256 reward) public
     ↪ onlyLosslessAdmin {

133 function setStakersReward(uint256 reward) public
     ↪ onlyLosslessAdmin {

140 function setCommitteeReward(uint256 reward) public
     ↪ onlyLosslessAdmin {

147 function setReportLifetime(uint256 lifetime) public
     ↪ onlyLosslessAdmin {

240 function reporterClaim(uint256 reportId) public whenNotPaused {

264 function retrieveCompensation(address adr, uint256 amount)
     ↪ public onlyLosslessGov {
```

3.11 CVF-11

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The argument type should be "ILssGovernance".

Listing 11:

```
105 function setLosslessGovernance(address _losslessGovernance)
    ↪ public onlyLosslessAdmin {
```

3.12 CVF-12

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessReporting.sol

Description This event is emitted even if nothing actually changed.

Listing 12:

```
114 emit ReportingAmountChanged(_reportingAmount);
```

3.13 CVF-13

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessReporting.sol

Recommendation The "0 <= reward" part is redundant, as "reward" is unsigned.

Listing 13:

```
120 require(reward + losslessReward + committeeReward +
    ↪ stakersReward <= 100 && 0 <= reward, "LSS: Total exceed
    ↪ 100");

127 require(reporterReward + reward + committeeReward +
    ↪ stakersReward <= 100 && 0 <= reward, "LSS: Total exceed
    ↪ 100");

134 require(reporterReward + losslessReward + committeeReward +
    ↪ reward <= 100 && 0 <= reward, "LSS: Total exceed 100");

141 require(reporterReward + losslessReward + reward + stakersReward
    ↪  <= 100 && 0 <= reward, "LSS: Total exceed 100");
```

3.14 CVF-14

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessReporting.sol

Description These checks are very similar.
Recommendation Consider moving them into a modifier that executed the function body and then ensures that the sum of the rewards doesn't exceed 100%.

Listing 14:

```
120 require(reward + losslessReward + committeeReward +
    ↪ stakersReward <= 100 && 0 <= reward, "LSS: Total exceed
    ↪ 100");

127 require(reporterReward + reward + committeeReward +
    ↪ stakersReward <= 100 && 0 <= reward, "LSS: Total exceed
    ↪ 100");

134 require(reporterReward + losslessReward + committeeReward +
    ↪ reward <= 100 && 0 <= reward, "LSS: Total exceed 100");

141 require(reporterReward + losslessReward + reward + stakersReward
    ↪  <= 100 && 0 <= reward, "LSS: Total exceed 100");
```

3.15 CVF-15

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The type of the "token" argument should be "ILERC20".

Listing 15:

```
176 function report(address token, address account) public
    ↪ notBlacklisted whenNotPaused returns (uint256){
```

3.16 CVF-16

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessReporting.sol

Description This variable is used without being initialized.
Recommendation Consider explicitly initializing to 0

Listing 16:

```
189 reportCount += 1;
```

3.17 CVF-17

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessReporting.sol

Description The returned value is ignored.

Listing 17:

```
197 stakingToken.transferFrom(msg.sender, address(this),
    ↪ reportingAmount);

247 ILERC20(reportTokens[reportId]).transfer(msg.sender,
    ↪ reporterClaimableAmount(reportId));
stakingToken.transfer(msg.sender, reportingAmount);

265 stakingToken.transfer(adr, amount);
```

3.18 CVF-18

- **Severity** Major
- **Status** Info
- **Category** Unclear Behaviour
- **Source** LosslessReporting.sol

Description The hacker may front run the original report with a transaction that sends stolen tokens to several different addresses, but a second report may blocklist only one of these addresses.
Recommendation Consider allowing several addresses in a second report.
Client Comment Emergency mode prevents hackers from sending the tokens after the initial report. Allowing to report more than two addresses could do lots of damage when used by malicious actors.

Listing 18:

```
226 require(secondReports[reportId] == false, "LSS: Another already
    ↪ submitted");
```

3.19 CVF-19

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessReporting.sol

Recommendation The "10**2" value should be a named constant.

Listing 19:

```
258 return reportedAmount * reporterReward / 10**2;
```

3.20 CVF-20

- **Severity** Moderate
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Recommendation This contract should implement the "ILssGovernance" interface to ensure consistency between the contract and the interface.

Listing 20:

```
17 contract LosslessGovernance is Initializable,
     ↪ AccessControlUpgradeable, PausableUpgradeable {
```

3.21 CVF-21

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessGovernance.sol

Description These variables are used without being initialized.
Recommendation Consider explicitly initializing them to 0.

Listing 21:

```
19 uint256 public lssTeamVoteIndex;

25 uint256 public committeeMembersCount;
```

3.22 CVF-22

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessGovernance.sol

Recommendation These variables should be turned into constants.

Listing 22:

```
20 uint256 public tokenOwnersVoteIndex;
   uint256 public committeeVoteIndex;
```

3.23 CVF-23

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Recommendation Role constants are usually declared public.

Listing 23:

```
23 bytes32 private constant COMMITTEE_ROLE = keccak256("
   ↪ COMMITTEE_ROLE");
```

3.24 CVF-24

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Recommendation It would be more efficient to replace these mappings with a single mapping whose keys are report IDs and values are structs encapsulating the values of the original structs.

Listing 24:

```
43 mapping(uint256 => Vote) public reportVotes;
   mapping(uint256 => uint256) public amountReported;
   mapping(uint256 => uint256) private retrievalAmount;

47 mapping(uint256 => ProposedWallet) public proposedWalletOnReport
   ↪ ;

49 mapping(uint256 => bool) public losslessPayed;
```

3.25 CVF-25

- **Severity** Moderate
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Description Indexed non-atomic event parameters works differently, as only thee hash for an actual value is logged.
Recommendation Consider making the parameter non-indexed or emitting a separate event for every added/removed address.

Listing 25:

```
73 event NewCommitteeMembers(address[] indexed members);
   event CommitteeMembersRemoved(address[] indexed members);
```

3.26 CVF-26

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessGovernance.sol

Recommendation Events are usually named via nouns such as "CommitteeMembersRemoval".

Listing 26:

```
74 event CommitteeMembersRemoved(address[] indexed members);
   event LosslessTeamVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event TokenOwnersVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event CommitteeMemberVoted(uint256 indexed reportId, address
      ↪ indexed member, bool indexed vote);
   event ReportResolved(uint256 indexed reportId, bool indexed
      ↪ resolution);
   event WalletProposed(uint256 indexed reportId, address indexed
      ↪ wallet);
80 event WalletRejected(uint256 indexed reportId, address indexed
      ↪ wallet);
   event FundsRetrieved(uint256 indexed reportId, address indexed
      ↪ wallet);
   event CompensationRetrieved(address indexed wallet);
   event LosslessClaimed(address indexed token, uint256 indexed
      ↪ reportID);
```

3.27 CVF-27

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Recommendation It would be more efficient to have a pair of events instead of a single event with a boolean parameter.

Listing 27:

```
75 event LosslessTeamVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event TokenOwnersVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event CommitteeMemberVoted(uint256 indexed reportId, address
      ↪ indexed member, bool indexed vote);
   event ReportResolved(uint256 indexed reportId, bool indexed
      ↪ resolution);
```

3.28 CVF-28

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Recommendation Thee events should include the "amount" parameter.

Listing 28:

```
81 event FundsRetrieved(uint256 indexed reportId, address indexed
   ↪ wallet);
event CompensationRetrieved(address indexed wallet);
event LosslessClaimed(address indexed token, uint256 indexed
   ↪ reportID);
```

3.29 CVF-29

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessGovernance.sol

Recommendation The argument types should be "ILssReporting", ILssController, ILssStaking, and ILERC20 respectively.

Listing 29:

```
85 function initialize(address _losslessReporting, address
   ↪ _losslessController, address _losslessStaking, address
   ↪ _stakingToken) public initializer {
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** LosslessGovernance.sol

Recommendation The value "7 days" should be a named constant.
Client Comment The 'walletDisputePeriod' can be changed and shouldn't be a constant.

Listing 30:

```
90 walletDisputePeriod = 7 days;
```

3.31 CVF-31

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Recommendation These function should emit some events.

Listing 31:

```
158 function setStakingToken(address _stakingToken) public
      ↪ onlyLosslessAdmin {

165 function setDisputePeriod(uint256 timeFrame) public
      ↪ onlyLosslessAdmin {
```

3.32 CVF-32

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessGovernance.sol

Recommendation The argument type should be "ILERC20".

Listing 32:

```
158 function setStakingToken(address _stakingToken) public
      ↪ onlyLosslessAdmin {
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LosslessGovernance.sol

Description The variable "i" is not explicitly initialized.
Recommendation Consider explicitly initializing to zero.
Client Comment Code was refactored, issues do not apply anymore.

Listing 33:

```
176 for(uint256 i; i < reportVote.committeeVotes.length; i++) {

339     for(uint256 i; i < reportedAddresses.length; i++) {

362 for(uint256 i; i < addresses.length; i++) {
```

3.34 CVF-34

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "reportVote.committeeVotes.length" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 34:

```
176 for(uint256 i; i < reportVote.committeeVotes.length; i++) {

185 if ((reportVote.committeeVotes.length - agreeCount) > (
    ↪ committeeMembersCount / 2)) {
```

3.35 CVF-35

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "committeeMembersCount / 2" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 35:

```
181 if (agreeCount > (committeeMembersCount / 2)) {

185 if ((reportVote.committeeVotes.length - agreeCount) > (
    ↪ committeeMembersCount / 2)) {
```

3.36 CVF-36

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Recommendation Right shift would be more efficient than division.

Listing 36:

```
181 if (agreeCount > (committeeMembersCount / 2)) {

185 if ((reportVote.committeeVotes.length - agreeCount) > (
    ↪ committeeMembersCount / 2)) {

446 if (proposedWalletOnReport[reportId].committeeDisagree <
    ↪ committeeMembersCount/2 ){
```

3.37 CVF-37

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessGovernance.sol

Recommendation Should be ‘else if‘.

Listing 37:

```
185 if ((reportVote.committeeVotes.length - agreeCount) > (
    ↪ committeeMembersCount / 2)) {
```

3.38 CVF-38

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessGovernance.sol

Recommendation Should be ‘else return‘.

Listing 38:

```
189 return (false, false);
```

3.39 CVF-39

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Recommendation Should be: require (committeeMembersCount >= members.length);

Listing 39:

```
208 require(committeeMembersCount != 0, "LSS: committee has no
    ↪ members");
```

3.40 CVF-40

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description This code repeats several times.
Recommendation Consider extracting to a function or modifier.
Client Comment Code refactored. Issue does not apply anymore.

Listing 40:

```
227 uint256 reportTimestamp = losslessReporting.reportTimestamps(
      ↪ reportId);
    require(reportTimestamp != 0 && reportTimestamp +
      ↪ losslessReporting.reportLifetime() > block.timestamp, "LSS
      ↪ : report is not valid");

247 uint256 reportTimestamp = losslessReporting.reportTimestamps(
      ↪ reportId);
    require(reportTimestamp != 0 && reportTimestamp +
      ↪ losslessReporting.reportLifetime() > block.timestamp, "LSS
      ↪ : report is not valid");

270 uint256 reportTimestamp = losslessReporting.reportTimestamps(
      ↪ reportId);
    require(reportTimestamp != 0 && reportTimestamp +
      ↪ losslessReporting.reportLifetime() > block.timestamp, "LSS
      ↪ : report is not valid");
```

3.41 CVF-41

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description There is already a local variable for "reportVotes[reportId]".
Recommendation Consider using it here.

Listing 41:

```
232 require(!reportVotes[reportId].voted[lssTeamVoteIndex], "LSS:
      ↪ LSS already voted.");
```

3.42 CVF-42

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Recommendation Unused code should be deleted.

Listing 42:

```
296 /*
        require(hasRole(COMMITTEE_ROLE, msg.sender)
                || msg.sender == losslessController.admin()
                || msg.sender == ILERC20(losslessReporting.
                  ↪ reportTokens(reportId)).admin(),
300             "LSS: Role cannot resolve."); */
```

3.43 CVF-43

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessGovernance.sol

Recommendation Should be ‘agreeCount‘.

Listing 43:

```
308 uint256 aggreeCount;
```

3.44 CVF-44

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessGovernance.sol

Recommendation Consider initializing to 0.

Listing 44:

```
308 uint256 aggreeCount;
    uint256 voteCount;
```

3.45 CVF-45

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LosslessGovernance.sol

Recommendation Consider initializing to false in the following 'if' clause.
Client Comment Code refactored. Issue does not apply anymore.

Listing 45:

```
320 bool expired;
```

3.46 CVF-46

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "addresses[i]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 46:

```
363 losslessController.resolvedNegatively(addresses[i]);
compensation[addresses[i]].amount += (reportingAmount *
    ↪ compensationAmount) / 10**2;
compensation[addresses[i]].payed = false;
```

3.47 CVF-47

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "(reportingAmount * compensationAmount) / 10**2" is calculated on every loop iteration.
Recommendation Consider calculating once and reusing.

Listing 47:

```
364 compensation[addresses[i]].amount += (reportingAmount *
    ↪ compensationAmount) / 10**2;
```

3.48 CVF-48

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "compensation[addresses[i]]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 48:

```
364 compensation[addresses[i]].amount += (reportingAmount *
    ↪ compensationAmount) / 10**2;
compensation[addresses[i]].payed = false;
```

3.49 CVF-49

- **Severity** Minor
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessGovernance.sol

Description This functions don't explicitly check that a report with the given ID does exist.
Recommendation Consider adding such explicit check.

Listing 49:

```
375 function proposeWallet(uint256 reportId, address wallet) public
    ↪ whenNotPaused {

395 function rejectWallet(uint256 reportId) public whenNotPaused {

426 function retrieveFunds(uint256 reportId) public whenNotPaused {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "proposedWalletOnReport[reportid]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 50:

```
381 require(proposedWalletOnReport[reportId].wallet == address(0), "
      ↪ LSS: Wallet already proposed.");

383 proposedWalletOnReport[reportId].wallet = wallet;
    proposedWalletOnReport[reportId].timestamp = block.timestamp;
    proposedWalletOnReport[reportId].losslessVote = true;
    proposedWalletOnReport[reportId].tokenOwnersVote = true;
    proposedWalletOnReport[reportId].walletAccepted = true;
```

3.51 CVF-51

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Recommendation This code could be simplified and optimized like this: if (hasRole(COMMITTEE_ROLE, msg.sender)) { ... } else if (msg.sender == losslessController.admin()) { ... } else if (msg.sender == ILERC20(losslessReporting.reportTokens[reportId].admin())) { ... } else revert ("LSS: Role cannot reject.");

Listing 51:

```
399 bool isMember = hasRole(COMMITTEE_ROLE, msg.sender);
400 bool isLosslessTeam = msg.sender == losslessController.admin();
    bool isTokenOwner = msg.sender == ILERC20(losslessReporting.
      ↪ reportTokens(reportId)).admin();

403 require(isMember || isLosslessTeam || isTokenOwner, "LSS: Role
      ↪ cannot reject.");

405 if (isMember) {

409 } else if (isLosslessTeam) {

413 } else {

417 }
```

3.52 CVF-52

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description This function allows rejecting the same proposal multiple times for non-committee members with an event logged every time.
Recommendation Consider logging an event only once the proposal is rejected.

Listing 52:

```
419     _determineProposedWallet(reportId);

442 function _determineProposedWallet(uint256 reportId) private
    ↪ returns(bool){

464     proposedWalletOnReport[reportId].status = false;
        proposedWalletOnReport[reportId].losslessVote = true;
        proposedWalletOnReport[reportId].losslessVoted = false;
        proposedWalletOnReport[reportId].tokenOwnersVote = true;
        proposedWalletOnReport[reportId].tokenOwnersVoted = false;
        proposedWalletOnReport[reportId].walletAccepted = false;
```

3.53 CVF-53

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description This check is weird as msg.sender may not be the zero address.
Recommendation Consider removing this check.
Client Comment Code refactored. Issue does not apply anymore.

Listing 53:

```
427 require(msg.sender != address(0), "LERC20: Cannot be zero
    ↪ address");
```

3.54 CVF-54

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessGovernance.sol

Description The returned value is ignored.

Listing 54:

```
435 ILERC20(losslessReporting.reportTokens(reportId)).transfer(msg.
    ↪ sender, retrievalAmount[reportId]);

502 ILERC20(token).transfer(msg.sender, compensationPerMember);

515 ILERC20(losslessReporting.reportTokens(reportId)).transfer(
    ↪ losslessController.admin(), amountToClaim);
```

3.55 CVF-55

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description This function is called only from one place, and the returned value is ignored there.
Recommendation Consider not returning any value.
Client Comment Code refactored. Issue does not apply anymore.

Listing 55:

```
442 function _determineProposedWallet(uint256 reportId) private
    ↪ returns(bool){
```

3.56 CVF-56

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The expression "proposedWalletOnReport[reportId] is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 56:

```
446 if (proposedWalletOnReport[reportId].committeeDisagree <
    ↪ committeeMembersCount/2 ){

450 if (proposedWalletOnReport[reportId].losslessVote) {

454 if (proposedWalletOnReport[reportId].tokenOwnersVote) {

462 proposedWalletOnReport[reportId].wallet = address(0);
    proposedWalletOnReport[reportId].timestamp = block.timestamp;
    proposedWalletOnReport[reportId].status = false;
    proposedWalletOnReport[reportId].losslessVote = true;
    proposedWalletOnReport[reportId].losslessVoted = false;
    proposedWalletOnReport[reportId].tokenOwnersVote = true;
    proposedWalletOnReport[reportId].tokenOwnersVoted = false;
    proposedWalletOnReport[reportId].walletAccepted = false;
```

3.57 CVF-57

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessGovernance.sol

Description The erroneously reporter address should be able to call this function in order to get the compensation. This could not be the case for contracts.
Recommendation Consider implementing some schema to allow contracts to be compensated for being erroneously reported.
Client Comment Will be addressed in the next release (Lossless V4).

Listing 57:

```
475 function retrieveCompensation() public whenNotPaused {
```

3.58 CVF-58

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessGovernance.sol

Description This function doesn't record payment, so the same committee member could claim the reward several times.
Recommendation Consider setting "reportVotes[reportId].committeeMemberVoted[msg.sender]" to false before sending the reward to the committee member.

Listing 58:

```
490 function claimCommitteeReward(uint256 reportId) public {
```

3.59 CVF-59

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessGovernance.sol

Description Different error messages are used for the same error.
Recommendation Consider using the same error message.

Listing 59:

```
492 require(isReportSolved(reportId), "LSS: Report is not solved.");
509 require(isReportSolved(reportId), "LSS: Report still open");
```

3.60 CVF-60

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessStaking.sol

Recommendation This contract should implement the "ILssStaking" interface to ensure consistency between the contract and the interface.

Listing 60:

```
16 contract LosslessStaking is Initializable, ContextUpgradeable,
    ↪ PausableUpgradeable {
```

3.61 CVF-61

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessStaking.sol

Description This field is not set in the initializer, so the initial value is zero, which allows staking for free.
Recommendation Consider accepting the initial staking amount as an initializer argument.

Listing 61:

```
34 uint256 public stakingAmount;
```

3.62 CVF-62

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessStaking.sol

Recommendation It would be more efficient to replace these mapping with a single mapping whose keys are report IDs and values are structs encapsulating the values of the original mappings.

Listing 62:

```
37 mapping(uint256 => address[]) public stakers;

39 mapping(uint256 => uint256) public totalStakedOnReport;

41 mapping(uint256 => uint256) public reportCoefficient;
```

3.63 CVF-63

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessStaking.sol

Recommendation Events are usually named via nouns such as “Stake”.

Listing 63:

```
43 event Staked(address indexed token, address indexed account,
     ↪ uint256 reportId);
   event StakerClaimed(address indexed staker, address indexed
     ↪ token, uint256 indexed reportID);
   event StakingAmountChanged(uint256 indexed newAmount);
```

3.64 CVF-64

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessStaking.sol

Recommendation The event should contain thee "amount" parameter to report the exact amount of the report token transferred.

Listing 64:

```
44 event StakerClaimed(address indexed staker, address indexed
   ↪ token, uint256 indexed reportID);
```

3.65 CVF-65

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessStaking.sol

Recommendation The argument types should be "ILssReporting" and "ILssController" respectively.

Listing 65:

```
47 function initialize(address _losslessReporting, address
   ↪ _losslessController) public initializer {
```

3.66 CVF-66

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessStaking.sol

Description In the former modifier "msg.sender" is at the right while in the latter it is at the "left".
Recommendation Consider being more consistent.

Listing 66:

```
55 require(losslessController.admin() == msg.sender, "LSS: Must be
   ↪ admin");

60 require(msg.sender == losslessController.pauseAdmin(), "LSS:
   ↪ Must be pauseAdmin");
```

3.67 CVF-67

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessStaking.sol

Recommendation The argument type should be "ILssReporting".

Listing 67:

```
84 function setLssReporting(address _losslessReporting) public
     ↪ onlyLosslessAdmin {
```

3.68 CVF-68

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessStaking.sol

Recommendation These functions should emit some events.

Listing 68:

```
84 function setLssReporting(address _losslessReporting) public
     ↪ onlyLosslessAdmin {

92 function setStakingToken(address _stakingToken) public
     ↪ onlyLosslessAdmin {

100 function setLosslessGovernance(address _losslessGovernance)
     ↪ public onlyLosslessAdmin {
```

3.69 CVF-69

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessStaking.sol

Recommendation The argument type should be "ILERC20".

Listing 69:

```
92 function setStakingToken(address _stakingToken) public
     ↪ onlyLosslessAdmin {
```

3.70 CVF-70

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessStaking.sol

Recommendation The argument type should be "ILssGovernance".

Listing 70:

```
100 function setLosslessGovernance(address _losslessGovernance)
    ↪ public onlyLosslessAdmin {
```

3.71 CVF-71

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessStaking.sol

Description There is not range check for the "stakingAmount" argument. One may set it to zero, for example.
Recommendation Consider adding appropriate checks.

Listing 71:

```
108 function setStakingAmount(uint256 _stakingAmount) public
    ↪ onlyLosslessAdmin {
```

3.72 CVF-72

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessStaking.sol

Description The expression "stakes[msg.sender].stakeInfoOnReport[reportId]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 72:

```
132 stakes[msg.sender].stakeInfoOnReport[reportId].timestamp = block
    ↪ .timestamp;
stakes[msg.sender].stakeInfoOnReport[reportId].coefficient =
    ↪ stakerCoefficient;
stakes[msg.sender].stakeInfoOnReport[reportId].staked = true;
```

3.73 CVF-73

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessStaking.sol

Description The returned value is ignored.

Listing 73:

```
138 stakingToken.transferFrom(msg.sender, address(this),
    ↪ stakingAmount);

169 ILERC20(losslessReporting.reportTokens(reportId)).transfer(msg.
    ↪ sender, stakerClaimableAmount(reportId));
170 stakingToken.transfer( msg.sender, stakingAmount);
```

3.74 CVF-74

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessStaking.sol

Recommendation The denominators should be named constants.

Listing 74:

```
153 uint256 amountDistributedToStakers = amountStakedOnReport *
    ↪ stakersReward / 10**2;

155 uint256 coefficientMultiplier = ((amountDistributedToStakers *
    ↪ 10**6) / reportCoefficient[reportId]);
    uint256 stakerAmountToClaim = (coefficientMultiplier *
    ↪ stakerCoefficient) / 10**6;
```

3.75 CVF-75

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessStaking.sol

Recommendation The denominators could be rendered as: 1e2 1e6

Listing 75:

```
153 uint256 amountDistributedToStakers = amountStakedOnReport *
      ↪ stakersReward / 10**2;

155 uint256 coefficientMultiplier = ((amountDistributedToStakers *
      ↪ 10**6) / reportCoefficient[reportId]);
    uint256 stakerAmountToClaim = (coefficientMultiplier *
      ↪ stakerCoefficient) / 10**6;
```

3.76 CVF-76

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessStaking.sol

Description In case the staking amount changed between a stake was made and claimed, the amount retuned could not be the same as the amount staked.
Recommendation Consider storing per-stake staking amount.

Listing 76:

```
170 stakingToken.transfer(msg.sender, stakingAmount);
```

3.77 CVF-77

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Description We did not review this file. Is it used? If not the import should be deleted.

Listing 77:

```
7 import "hardhat/console.sol";
```

3.78 CVF-78

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Recommendation This interface should be defined in a separate file named "ProtectionStrategy.sol".

Listing 78:

```
14 interface ProtectionStrategy {
```

3.79 CVF-79

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The type of the "token" argument should be "IERC20".

Listing 79:

```
15  function isTransferAllowed(address token, address sender,
      ↪ address recipient, uint256 amount) external;

278 function proposeNewSettlementPeriod(address token, uint256
      ↪ _seconds) public {

289 function executeNewSettlementPeriod(address token) public {

304 function activateEmergency(address token) external
      ↪ onlyLosslessEnv {

315 function deactivateEmergency(address token) external
      ↪ onlyLosslessEnv {

331 function setProtectedAddress(address token, address
      ↪ protectedAddresss, ProtectionStrategy strategy) external
      ↪ onlyGuardian whenNotPaused {

341 function removeProtectedAddress(address token, address
      ↪ protectedAddresss) external onlyGuardian whenNotPaused {

350 function getLockedAmount(address token, address account) public
      ↪ view returns (uint256) {

370 function getAvailableAmount(address token, address account)
      ↪ public view returns (uint256 amount) {

397 function retrieveBlacklistedFunds(address[] calldata _addresses,
      ↪  address token, uint256 reportId) public onlyLosslessEnv
      ↪ returns(uint256){
```

3.80 CVF-80

- **Severity** Major
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Recommendation This contract should implement the "ILssController" interface to ensure consistency between the contract and the interface.

Listing 80:

```
20 contract LosslessControllerV3 is Initializable,
     ↪ ContextUpgradeable, PausableUpgradeable {
```

3.81 CVF-81

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The keys type should be "ILERC20".

Listing 81:

```
28 mapping(address => Protections) private tokenProtections;

45 mapping(address => uint256) public tokenLockTimeframe;
   mapping(address => uint256) public proposedTokenLockTimeframe;
   mapping(address => uint256) public changeSettlementTimelock;
   mapping(address => bool) public isNewSettlementProposed;

73 mapping(address => TokenLockedFunds) private
     ↪ tokenScopedLockedFunds;

79 mapping(address => EmergencyMode) private emergencyMode;

90 mapping (address => PeriodTransfers) private
     ↪ tokenTransferInPeriod;
```

3.82 CVF-82

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Recommendation These mappings could be merged into a single mapping whose keys are tokens and values are structs encapsulating the values of the original mappings.

Listing 82:

```
28 mapping(address => Protections) private tokenProtections;

45 mapping(address => uint256) public tokenLockTimeframe;
   mapping(address => uint256) public proposedTokenLockTimeframe;
   mapping(address => uint256) public changeSettlementTimelock;
   mapping(address => bool) public isNewSettlementProposed;

73 mapping(address => TokenLockedFunds) private
   ↪ tokenScopedLockedFunds;

79 mapping(address => EmergencyMode) private emergencyMode;

90 mapping (address => PeriodTransfers) private
   ↪ tokenTransferInPeriod;
```

3.83 CVF-83

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description This mapping is redundant. A special value in the "proposedTokenLockTimeframe" mapping could be used as an indicator of no proposal.

Listing 83:

```
48 mapping(address => bool) public isNewSettlementProposed;
```

3.84 CVF-84

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LosslessControllerV3.sol

Description It is unclear what this "erroneous compensation" means.
Recommendation Consider documenting.
Client Comment We've changed the variable name to compensationPercentage. When a report turns out to be invalid, the reported address is entitled to compensation, because he was blacklisted for a period of time by mistake.

Listing 84:

```
50 uint256 public erroneousCompensation;
```

3.85 CVF-85

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LosslessControllerV3.sol

Description Is it required that the staking token is a lossless token itself?
Client Comment It is expected that staking tokens will be lossless tokens, but for some chains where lossless token is not bridged yet, another token can be used.

Listing 85:

```
52 ILERC20 public stakingToken;
```

3.86 CVF-86

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** LosslessControllerV3.sol

Description It seems that an address can be both in black- and whitelist. If it is not supposed to happen, consider adding a check to the relevant functions.

Listing 86:

```
76 mapping(address => bool) public whitelist;
   mapping(address => bool) public blacklist;
```

3.87 CVF-87

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Recommendation This flag is redundant as it can be replaced by ‘emergencyTimestamp!=0‘.

Listing 87:

```
82 bool emergency;
```

3.88 CVF-88

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessControllerV3.sol

Recommendation Events are usually named via nouns, such as “AdminChange” or just “Admin”.

Listing 88:

```
92 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
   event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
   event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);

98 event GuardianSet(address indexed oldGuardian, address indexed
   ↪ newGuardian);
   event ProtectedAddressSet(address indexed token, address indexed
   ↪  protectedAddress, address indexed strategy);
100 event RemovedProtectedAddress(address indexed token, address
   ↪ indexed protectedAddress);
   event NewSettlementPeriodProposed(address token, uint256
   ↪ _seconds);
   event SettlementPeriodChanged(address token, uint256
   ↪ proposedTokenLockTimeframe);
```

3.89 CVF-89

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The type of the "token" parameters should be "ILERC20".

Listing 89:

```
99  event ProtectedAddressSet(address indexed token, address indexed
    ↪  protectedAddress, address indexed strategy);
100 event RemovedProtectedAddress(address indexed token, address
    ↪ indexed protectedAddress);
    event NewSettlementPeriodProposed(address token, uint256
    ↪ _seconds);
    event SettlementPeriodChanged(address token, uint256
    ↪ proposedTokenLockTimeframe);
```

3.90 CVF-90

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Recommendation The "token" parameters should be indexed.

Listing 90:

```
101 event NewSettlementPeriodProposed(address token, uint256
    ↪ _seconds);
    event SettlementPeriodChanged(address token, uint256
    ↪ proposedTokenLockTimeframe);
```

3.91 CVF-91

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Description In other similar modifiers, "msg.sender" is at the left, while here it is at the right.
Recommendation Consider being more consistent.

Listing 91:

```
114 require(admin == msg.sender, "LSS: Must be admin");
```

3.92 CVF-92

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description These checks are weird, as 'msg.sender' cannot be the zero address.
Recommendation Consider removing these checks.

Listing 92:

```
163 require(msg.sender != address(0), "LERC20: Cannot be zero
    ↪ address");

169 require(msg.sender != address(0), "LERC20: Cannot be zero
    ↪ address");

175 require(msg.sender != address(0), "LERC20: Cannot be zero
    ↪ address");
```

3.93 CVF-93

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description These events are emitted even if nothing actually changed.

Listing 93:

```
164 emit AdminChanged(admin, newAdmin);

170 emit RecoveryAdminChanged(recoveryAdmin, newRecoveryAdmin);

176 emit PauseAdminChanged(pauseAdmin, newPauseAdmin);
```

3.94 CVF-94

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The argument type should be "ILERC20".
Client Comment Code refactored. Issue does not apply anymore.

Listing 94:

```
187 function setStakingToken(address _stakingToken) public
    ↪ onlyLosslessAdmin {
```

3.95 CVF-95

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Recommendation These functions should emit some events.

Listing 95:

```
187 function setStakingToken(address _stakingToken) public
      ↪ onlyLosslessAdmin {

195 function setSettlementTimeLock(uint256 newTimelock) public
      ↪ onlyLosslessAdmin {

202 function setDexTrasnferThreshold(uint256 newThreshold) public
      ↪ onlyLosslessAdmin {

208 function setCompensationAmount(uint256 amount) public
      ↪ onlyLosslessAdmin {

215 function setLocksLiftUpExpiration(uint256 time) public
      ↪ onlyLosslessAdmin {

222 function setDexList(address[] calldata _dexList, bool value)
      ↪ public onlyLosslessAdmin {

231 function setWhitelist(address[] calldata _addrList, bool value)
      ↪ public onlyLosslessAdmin {

241 function addToBlacklist(address _adr) public onlyLosslessEnv {

248 function resolvedNegatively(address _adr) public onlyLosslessEnv
      ↪  {

255 function setStakingContractAddress(address _adr) public
      ↪ onlyLosslessAdmin {

262 function setReportingContractAddress(address _adr) public
      ↪ onlyLosslessAdmin {

269 function setGovernanceContractAddress(address _adr) public
      ↪ onlyLosslessAdmin {

304 function activateEmergency(address token) external
      ↪ onlyLosslessEnv {

315 function deactivateEmergency(address token) external
      ↪ onlyLosslessEnv {
```

3.96 CVF-96

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LosslessControllerV3.sol

Recommendation Should be 'Transfer'.

Listing 96:

```
202 function setDexTrasnferThreshold(uint256 newThreshold) public
    ↪ onlyLosslessAdmin {
```

3.97 CVF-97

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description It seems that only an integer number of percents could be specified.
Recommendation Consider supporting better precision.
Client Comment Code refactored. Issue does not apply anymore.

Listing 97:

```
209 require(0 <= amount && amount <= 100, "LSS: Invalid amount");
```

3.98 CVF-98

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The argument type should be "ILssStaking".

Listing 98:

```
255 function setStakingContractAddress(address _adr) public
    ↪ onlyLosslessAdmin {
```

3.99 CVF-99

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The argument type should be "ILssReporting".

Listing 99:

```
262 function setReportingContractAddress(address _adr) public
    ↪ onlyLosslessAdmin {
```

3.100 CVF-100

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The argument type should be "ILssGovernance".

Listing 100:

```
269 function setGovernanceContractAddress(address _adr) public
    ↪ onlyLosslessAdmin {
```

3.101 CVF-101

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessControllerV3.sol

Recommendation This function should clear the "proposedTokenLockTimeframe[token]" value.

Listing 101:

```
289 function executeNewSettlementPeriod(address token) public {
```

3.102 CVF-102

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The "proposedTokenLockTimeframe[token]" expression is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 102:

```
293 tokenLockTimeframe[token] = proposedTokenLockTimeframe[token];

295 emit SettlementPeriodChanged(token, proposedTokenLockTimeframe[
    ↪ token]);
```

3.103 CVF-103

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The expression "emergencyMode[token]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 103:

```
305 emergencyMode[token].emergency = true;
emergencyMode[token].emergencyTimestamp = block.timestamp;
```

3.104 CVF-104

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LosslessControllerV3.sol

Recommendation Should be 'deactivates'.

Listing 104:

```
309 /// @notice This function activates the emergency mode
```

3.105 CVF-105

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** LosslessControllerV3.sol

Recommendation This function should set "emergencyMode[token].emergency" to false.

Listing 105:

```
315 function deactivateEmergency(address token) external
    ↪ onlyLosslessEnv {
```

3.106 CVF-106

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LosslessControllerV3.sol

Recommendation Should be 'Address'.

Listing 106:

```
331 function setProtectedAddress(address token, address
    ↪ protectedAddresss, ProtectionStrategy strategy) external
    ↪ onlyGuardian whenNotPaused {
```

3.107 CVF-107

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Recommendation This event is emitted even if no protection was set.

Listing 107:

```
343 emit RemovedProtectedAddress(token, protectedAddresss);
```

3.108 CVF-108

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description This function doesn't scale. In case of too many queued locks, it could exceed the block gas limit.
Recommendation Consider optimizing.

Listing 108:

```
350 function getLockedAmount(address token, address account) public
    ↪ view returns (uint256) {
```

3.109 CVF-109

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessControllerV3.sol

Recommendation Consider initializing to 0.

Listing 109:

```
351 uint256 lockedAmount;
```

3.110 CVF-110

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The expression "queue.last" is calculated on every loop iteration.
Recommendation Consider calculating once and reusing. For example, store the cumulatinve locked amount in the queue, i.e. the sum of the locked amounts up to a checkpoint. Also, store the current cumulative locked amount, i.e. the sum of all the locked amounts up to the current time. For a particular checkpoint, the different between the current cumulative locked amount and the checkpoint's cumulative locked amount is the amount locked after the ckechpoint. Use binary serach to find the newest checkpoint whose tokens are already unlocked, then calculated the different to find out the locked amount.

Listing 110:

```
357 while (i <= queue.last) {
```

3.111 CVF-111

- **Severity** Minor
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** LosslessControllerV3.sol

Description An overflow in a single checkpoint might revert the entire loop.
Recommendation Consider explaining why overflow is not possible here.

Listing 111:

```
360 lockedAmount = lockedAmount + checkpoint.amount;
```

3.112 CVF-112

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Recommendation This function doesn't scale because the "getLockedAmount" function doesn't scale.

Listing 112:

```
370 function getAvailableAmount(address token, address account)
    ↪ public view returns (uint256 amount) {
```

3.113 CVF-113

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The "queue.last" value is read from the storage several times.
Recommendation Consider reading once and reusing.

Listing 113:

```
385 if (queue.lockedFunds[queue.last].timestamp == checkpoint.
    ↪ timestamp) {
      queue.lockedFunds[queue.last].amount += checkpoint.amount;

388       queue.last += 1;
```

3.114 CVF-114

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description Here the "queue.last" value, that was just written into the storage, is read back.
Recommendation Consider reusing the written value.

Listing 114:

```
389 queue.lockedFunds[queue.last] = checkpoint;
```

3.115 CVF-115

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LosslessControllerV3.sol

Recommendation The "10**2" value could be rendered as "1e2".

Listing 115:

```
405 uint256 toLssStaking = totalAmount * (stakersReward +
      ↪ losslessReward) / 10**2;
    uint256 toLssReporting = totalAmount * reporterReward / 10**2;

413 return totalAmount - toLssStaking - toLssReporting - (
      ↪ totalAmount * committeeReward / 10**2);
```

3.116 CVF-116

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV3.sol

Recommendation The "10**2" value should be a named constant.

Listing 116:

```
405 uint256 toLssStaking = totalAmount * (stakersReward +
      ↪ losslessReward) / 10**2;
    uint256 toLssReporting = totalAmount * reporterReward / 10**2;

413 return totalAmount - toLssStaking - toLssReporting - (
      ↪ totalAmount * committeeReward / 10**2);
```

3.117 CVF-117

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The denominator used is quiet small and thus precision is quite low.
Recommendation Consider using a bigger denominator.
Client Comment This precision is acceptable.

Listing 117:

```
405 uint256 toLssStaking = totalAmount * (stakersReward +
      ↪ losslessReward) / 10**2;
    uint256 toLssReporting = totalAmount * reporterReward / 10**2;

413 return totalAmount - toLssStaking - toLssReporting - (
      ↪ totalAmount * committeeReward / 10**2);
```

3.118 CVF-118

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessControllerV3.sol

Description The returned value is ignored.

Listing 118:

```
409 ILERC20(token).transfer(address(losslessStaking), toLssStaking);
410 ILERC20(token).transfer(address(losslessReporting),
      ↪ toLssReporting);
    ILERC20(token).transfer(address(losslessGovernance),
      ↪ toLssGovernance);
```

3.119 CVF-119

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LosslessControllerV3.sol

Description The comment seems irrelevant.
Recommendation Consider rewriting it.

Listing 119:

```
419 /// @param availableAmount Address to lift the locks

421 /// @param amount Address to lift the locks
```

3.120 CVF-120

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description This function may return ‘amountLeft‘.
Client Comment Doesn’t seem necessary.

Listing 120:

```
422 function _removeUsedUpLocks (uint256 availableAmount, address
    ↪ account, uint256 amount) private {
```

3.121 CVF-121

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** LosslessControllerV3.sol

Recommendation Should be “queue.first”.

Listing 121:

```
429 uint256 i = 1;
```

3.122 CVF-122

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** LosslessControllerV3.sol

Description This condition assumes that the timestamps in the queue go in ascending order, while this could not be the case, as the “tokenLockTimeFrame” value for an account could be changed.
Recommendation Consider enforcing the timestamp in the queue to go in ascending order.

Listing 122:

```
455 while (checkpoint.timestamp <= block.timestamp && i <= queue.
    ↪ last) {
```

3.123 CVF-123

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description This could load a non-existing checkpoint.
Recommendation Consider refactoring the loop.
Client Comment Code refactored. Issue does not apply anymore.

Listing 123:

```
458 checkpoint = queue.lockedFunds[i];
```

3.124 CVF-124

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description This condition ignores the "emergencyMode[msg.sender].emergency" flag.
Recommendation Consider taking this flag into account.
Client Comment Code refactored. Issue does not apply anymore.

Listing 124:

```
472 require(emergencyMode[msg.sender].emergencyTimestamp +
    ↪ tokenLockTimeframe[msg.sender] < block.timestamp,
```

3.125 CVF-125

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV3.sol

Description The "msgSender" value is not passed to the strategy.
Recommendation Consider passing it.
Client Comment Protocol allows transferring tokens on behalf of other accounts without checks.

Listing 125:

```
511 tokenProtections[msg.sender].protections[sender].strategy.
    ↪ isTransferAllowed(msg.sender, sender, recipient, amount);
```

3.126 CVF-126

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV3.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 126:

```
524 function beforeMint(address to, uint256 amount) external {}

526 function beforeApprove(address sender, address spender, uint256
      ↪ amount) external {}

528 function beforeBurn(address account, uint256 amount) external {}

530 function beforeIncreaseAllowance(address msgSender, address
      ↪ spender, uint256 addedValue) external {}

532 function beforeDecreaseAllowance(address msgSender, address
      ↪ spender, uint256 subtractedValue) external {}

539 function afterMint(address to, uint256 amount) external {}

541 function afterApprove(address sender, address spender, uint256
      ↪ amount) external {}

543 function afterBurn(address account, uint256 amount) external {}

545 function afterTransfer(address sender, address recipient,
      ↪ uint256 amount) external {}

547 function afterTransferFrom(address msgSender, address sender,
      ↪ address recipient, uint256 amount) external {}

549 function afterIncreaseAllowance(address sender, address spender,
      ↪  uint256 addedValue) external {}

551 function afterDecreaseAllowance(address sender, address spender,
      ↪  uint256 subtractedValue) external {}
```

3.127 CVF-127

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV2.sol

Recommendation This interface should be defined in a separate file named "ProtectionStrategy.sol".

Listing 127:

```
8 interface ProtectionStrategy {
```

3.128 CVF-128

- **Severity** Minor
- **Status** Info
- **Category** Bad naming
- **Source** LosslessControllerV2.sol

Description The function name suggests that it should return a boolean value, while it actually returns nothing.
Recommendation Consider renaming to "checkTransferAllowed".
Client Comment Suggestion makes sense, but all contracts in vault-protection are already deployed and used, so we won't be implementing breaking changes on those.

Listing 128:

```
9 function isTransferAllowed(address token, address sender,
    ↪ address recipient, uint256 amount) external;
```

3.129 CVF-129

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** LosslessControllerV2.sol

Recommendation The type of the "token" argument should be "IERC20".
Client Comment Suggestion makes sense, but all contracts in vault-protection are already deployed and used, so we won't be implementing changes on those.

Listing 129:

```
 9 function isTransferAllowed(address token, address sender,
     ↪ address recipient, uint256 amount) external;

76 function isAddressProtected(address token, address
     ↪ protectedAddress) external view returns (bool) {

80 function getProtectedAddressStrategy(address token, address
     ↪ protectedAddress) external view returns (address) {

124 function setProtectedAddress(address token, address
     ↪ protectedAddresss, ProtectionStrategy strategy) external
     ↪ onlyGuardian whenNotPaused {

134 function removeProtectedAddress(address token, address
     ↪ protectedAddresss) external onlyGuardian whenNotPaused {
```

3.130 CVF-130

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LosslessControllerV2.sol

Description These variables are used without being initialized.
Recommendation Consider explicitly initializing them with 0.
Client Comment They are initialised in V1, V2 has the same state because these contracts use upgradable proxy pattern.

Listing 130:

```
13 address public pauseAdmin;
   address public admin;
   address public recoveryAdmin;

19 address public guardian;
```

3.131 CVF-131

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV2.sol

Recommendation The key type should be "IERC20".

Listing 131:

```
20 mapping(address => Protections) private tokenProtections;
```

3.132 CVF-132

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessControllerV2.sol

Recommendation Events are usually named via nouns, such as "AdminChange" or just "Admin".
Client Comment Partially fixed. We won't be renaming events that are already used in production.

Listing 132:

```
33 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
   event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
   event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);

40 event GuardianSet(address indexed oldGuardian, address indexed
   ↪ newGuardian);
   event ProtectedAddressSet(address indexed token, address indexed
   ↪  protectedAddress, address indexed strategy);
   event RemovedProtectedAddress(address indexed token, address
   ↪ indexed protectedAddress);
```

3.133 CVF-133

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV2.sol

Recommendation The previous value parameters are redundant as their values could be derived from the previous events.
Client Comment We won't be implementing this, because it can break our current production system that relies on these events.

Listing 133:

```
33 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
   event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
   event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);

40 event GuardianSet(address indexed oldGuardian, address indexed
   ↪ newGuardian);
```

3.134 CVF-134

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LosslessControllerV2.sol

Recommendation The type of the "token" parameters should be "IERC20".

Listing 134:

```
41 event ProtectedAddressSet(address indexed token, address indexed
   ↪  protectedAddress, address indexed strategy);
   event RemovedProtectedAddress(address indexed token, address
   ↪ indexed protectedAddress);
```

3.135 CVF-135

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV2.sol

Description In other similar modifiers, "msg.sender" is at the left, while here is it at the right.
Recommendation Consider being more consistent.

Listing 135:

```
52 require(admin == msg.sender, "LOSSLESS: Must be admin");
```

3.136 CVF-136

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** LosslessControllerV2.sol

Description For an unprotected address this function silently returns some address, most probably zero.
Recommendation Consider reverting in such a case.
Client Comment That's expected behaviour.

Listing 136:

```
80 function getProtectedAddressStrategy(address token, address
   ↪ protectedAddress) external view returns (address) {
```

3.137 CVF-137

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV2.sol

Description Transferring an ownership to the zero address is quite a common way to make a contract unowned.
Recommendation Consider removing these checks, as the current owner may still transfer the ownership to a dead address.

Listing 137:

```
95  require(newAdmin != address(0), "LERC20: Cannot be zero address
    ↪ ");

101 require(newRecoveryAdmin != address(0), "LERC20: Cannot be zero
    ↪ address");

107 require(newPauseAdmin != address(0), "LERC20: Cannot be zero
    ↪ address");
```

3.138 CVF-138

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV2.sol

Description These events are emitted even if nothing actually changed.

Listing 138:

```
96  emit AdminChanged(admin, newAdmin);

102 emit RecoveryAdminChanged(recoveryAdmin, newRecoveryAdmin);

108 emit PauseAdminChanged(pauseAdmin, newPauseAdmin);

117 emit GuardianSet(guardian, newGuardian);

128 emit ProtectedAddressSet(token, protectedAddresss, address(
    ↪ strategy));

136 emit RemovedProtectedAddress(token, protectedAddresss);
```

3.139 CVF-139

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LosslessControllerV2.sol

Description The value of the "msgSender" argument is ignored.
Recommendation Consider passing it to the strategy.
Client Comment Protocol allows transferring tokens on behalf of other accounts without checks.

Listing 139:

```
151 function beforeTransferFrom(address msgSender, address sender,
    ↪ address recipient, uint256 amount) external {
```

3.140 CVF-140

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV2.sol

Recommendation it is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 140:

```
157 function beforeApprove(address sender, address spender, uint256
    ↪ amount) external {}

159 function beforeIncreaseAllowance(address msgSender, address
    ↪ spender, uint256 addedValue) external {}

161 function beforeDecreaseAllowance(address msgSender, address
    ↪ spender, uint256 subtractedValue) external {}

167 function afterApprove(address sender, address spender, uint256
    ↪ amount) external {}

169 function afterTransfer(address sender, address recipient,
    ↪ uint256 amount) external {}

171 function afterTransferFrom(address msgSender, address sender,
    ↪ address recipient, uint256 amount) external {}

173 function afterIncreaseAllowance(address sender, address spender,
    ↪  uint256 addedValue) external {}

175 function afterDecreaseAllowance(address sender, address spender,
    ↪  uint256 subtractedValue) external {}
```

3.141 CVF-141

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation This contract should be moved to a separate file named "Context.sol".

Listing 141:

```
4 abstract contract Context {
```

3.142 CVF-142

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation This line is redundant. Even without it Solidity compiler doesn't report any warnings.

Listing 142:

```
10 this; // silence state mutability warning without generating
     ↪ bytecode — see https://github.com/ethereum/solidity/issues
     ↪ /2691
```

3.143 CVF-143

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation This interface should be moved to a separate file named "IERC20".

Listing 143:

```
15 interface IERC20 {
```

3.144 CVF-144

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation This interface should be moved to a file named "ILosslessController".

Listing 144:

```
33 interface ILosslessController {
```

3.145 CVF-145

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation These three functions could be replaced with a single one: function beforeApprove (address sender, address spender, uint256 oldAllowance, uint256 newAllowance) external;
Client Comment These contracts are already deployed and widely used, we cannot introduce any interface changes on LosslessController as that would brick legacy tokens.

Listing 145:

```
38 function beforeApprove(address sender, address spender, uint256
   ↪ amount) external;

40 function beforeIncreaseAllowance(address msgSender, address
   ↪ spender, uint256 addedValue) external;

42 function beforeDecreaseAllowance(address msgSender, address
   ↪ spender, uint256 subtractedValue) external;
```

3.146 CVF-146

- **Severity** Major
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation This contract should implement the "ILERC20" interface to ensure consistency between the contract and the interface.

Listing 146:

```
45 contract LERC20 is Context, IERC20 {
```

3.147 CVF-147

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LERC20.sol

Recommendation Should be 'candidate'.

Listing 147:

```
53 address private recoveryAdminCanditate;
```

3.148 CVF-148

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation This variable is redundant. Just set the "losslessTurnOffTimestamp" to "type(uint256).max" to effectively make a turn off not proposed.

Listing 148:

```
58 bool public isLosslessTurnOffProposed;
```

3.149 CVF-149

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation These events should be moved to the "ILERC20" interface.

Listing 149:

```
62 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event RecoveryAdminChangeProposed(address indexed candidate);
event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
event LosslessTurnOffProposed(uint256 turnOffDate);
event LosslessTurnedOff();
event LosslessTurnedOn();
```

3.150 CVF-150

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation The previous value parameters are redundant as they could be derived from the previous events.

Listing 150:

```
62 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);

64 event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
```

3.151 CVF-151

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LERC20.sol

Recommendation Events are usually named via nouns such as "AdminChange" or just "Admin", "RecoveryAdminChange" or just "RecoveryAdmin" etc.

Listing 151:

```
62 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event RecoveryAdminChangeProposed(address indexed candidate);
event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
event LosslessTurnOffProposed(uint256 turnOffDate);
event LosslessTurnedOff();
event LosslessTurnedOn();
```

3.152 CVF-152

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation This function is redundant as the "admin" variable is already public.
Client Comment This function is used by the guardian contract which is already deployed and used in production, so we won't be changing this at the moment.

Listing 152:

```
123 function getAdmin() external view returns (address) {
```

3.153 CVF-153

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This function performs several independent transfers.
Recommendation It could be optimized like this: uint256 fromLength = from.length; uint256 totalAmount = 0; for (uint256 i = 0; i < fromLength; i++) { address fromAddress = from[i]; uint256 fromBalance = _balances [fromAddress]; _balances [fromAddress] = 0; amount += fromBalance; emit Transfer (fromAddress, address (lossless), fromAmount); } _balances [address (lossless)] += amount;

Listing 153:

```
127 function transferOutBlacklistedFunds(address[] calldata from)
    ↪ external {
```

3.154 CVF-154

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description The expression "from[i]" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 154:

```
130 _transfer(from[i], address(lossless), balanceOf(from[i]));
```

3.155 CVF-155

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation This check is redundant. It is anyway possible to set an invalid admin address.

Listing 155:

```
135 require(newAdmin != address(0), "LERC20: Cannot be zero address
    ↪ ");
```

3.156 CVF-156

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This event is emitted even if nothing actually changed.

Listing 156:

```
136 emit AdminChanged(admin, newAdmin);
```

3.157 CVF-157

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This check doesn't allow cancelling a pending ownership transfer without proposing a new candidate.
Recommendation Consider removing this check or adding a separate function to cancel a pending ownership transfer.

Listing 157:

```
141 require(candidate != address(0), "LERC20: Cannot be zero address
    ↪ ");
```

3.158 CVF-158

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation This function should clear the "recoveryAdminCandidate" variable.

Listing 158:

```
147 function acceptRecoveryAdminOwnership(bytes memory key) external
    ↪ {
```

3.159 CVF-159

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LERC20.sol

Description These values are used without being initialized.
Recommendation Consider explicitly initializing to 0.

Listing 159:

```
148 require(_msgSender() == recoveryAdminCanditate, "LERC20: Must be
    ↪ canditate");
    require(keccak256(key) == recoveryAdminKeyHash, "LERC20: Invalid
    ↪ key");
```

3.160 CVF-160

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LERC20.sol

Description This check looks redundant. What problem is it supposed to solve?
Client Comment This ensures that when ownership is transferred to an address by mistake, this new address is not able to claim ownership without knowing the key. The key has to be shared off chain.

Listing 160:

```
149 require(keccak256(key) == recoveryAdminKeyHash, "LERC20: Invalid
    ↪ key");
```

3.161 CVF-161

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This function could be used not only to propose a lossless turn off, but also to postpone execution of an existing lossless turn off proposal.
Recommendation Consider doing nothing in case a lossless turn off is already proposed.

Listing 161:

```
154 function proposeLosslessTurnOff() external onlyRecoveryAdmin {
```

3.162 CVF-162

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This function should revert in case lossless is currently not turned on.

Listing 162:

```
154 function proposeLosslessTurnOff() external onlyRecoveryAdmin {
```

3.163 CVF-163

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description These events are emitted even if nothing actually changed.

Listing 163:

```
165 emit LosslessTurnedOff();

171 emit LosslessTurnedOn();
```

3.164 CVF-164

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description This check makes the contract non-complient with ERC-20 and could cause problems with certain DeFi protocols that don't set allowance to zero before setting it back to a non-zero value.
Recommendation Consider removing this check.

Listing 164:

```
206 require((amount == 0) || (_allowances[_msgSender()][spender] ==
    ↪ 0), "LERC20: Cannot change non zero allowance");
```

3.165 CVF-165

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LERC20.sol

Recommendation It would be better to do this check before the transfer.

Listing 165:

```
215 require(currentAllowance >= amount, "LERC20: transfer amount
    ↪ exceeds allowance");
```

3.166 CVF-166

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LERC20.sol

Recommendation This emits am "Approval" event which is not 100% compliant with the ERC-20 standard.
Client Comment This is acceptable.

Listing 166:

```
216 _approve(sender, _msgSender(), currentAllowance - amount);
```

3.167 CVF-167

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Description Transferring tokens to the zero address is quite a common way to destroy the tokens.
Recommendation Consider not forbidding this.

Listing 167:

```
236 require(recipient != address(0), "LERC20: transfer to the zero
    ↪ address");
```

3.168 CVF-168

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LERC20.sol

Recommendation These checks are redundant and just waste gas.

Listing 168:

```
255 require(owner != address(0), "LERC20: approve from the zero
    ↪ address");
    require(spender != address(0), "LERC20: approve to the zero
    ↪ address");
```

3.169 CVF-169

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LosslessControllerV1.sol

Recommendation Events are usually named via nouns, such as "AdminChange" or simply "Admin".

Listing 169:

```
13 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
   event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
   event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
```

3.170 CVF-170

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV1.sol

Recommendation The previous value parameters are redundant as their values could be derived from the previous events.

Listing 170:

```
13 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
   event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
   event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
```

3.171 CVF-171

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LosslessControllerV1.sol

Description It is confusing the "_msgSender()" is at the left in the former case and at the right in the latter case.
Recommendation Consider being more consistent.
Client Comment Not fixing this, because this contract is not relevant anymore v2 is the one that's in production at the moment.

Listing 171:

```
20 require(_msgSender() == recoveryAdmin, "LOSSLESS: Must be
   ↪ recoveryAdmin");

25 require(admin == _msgSender(), "LOSSLESS: Must be admin");
```

3.172 CVF-172

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV1.sol

Recommendation This function should call the following functions: __Context_init_unchained __Pausable_init_unchained

Listing 172:

```
29 function initialize(address _admin, address _recoveryAdmin,
   ↪ address _pauseAdmin) public initializer {
```

3.173 CVF-173

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LosslessControllerV1.sol

Description These events are emitted even if nothing actually changed.

Listing 173:

```
48 emit AdminChanged(admin, newAdmin);

53 emit RecoveryAdminChanged(recoveryAdmin, newRecoveryAdmin);

58 emit PauseAdminChanged(pauseAdmin, newPauseAdmin);
```

3.174 CVF-174

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LosslessControllerV1.sol

Recommendation it is a good practice to put a comment into an empty block to explain why the block is empty.

Listing 174:

```
70 function beforeTransfer(address sender, address recipient,
     ↪ uint256 amount) external {}

72 function beforeTransferFrom(address msgSender, address sender,
     ↪ address recipient, uint256 amount) external {}

74 function beforeApprove(address sender, address spender, uint256
     ↪ amount) external {}

76 function beforeIncreaseAllowance(address msgSender, address
     ↪ spender, uint256 addedValue) external {}

78 function beforeDecreaseAllowance(address msgSender, address
     ↪ spender, uint256 subtractedValue) external {}

82 function afterApprove(address sender, address spender, uint256
     ↪ amount) external {}

84 function afterTransfer(address sender, address recipient,
     ↪ uint256 amount) external {}

86 function afterTransferFrom(address msgSender, address sender,
     ↪ address recipient, uint256 amount) external {}

88 function afterIncreaseAllowance(address sender, address spender,
     ↪  uint256 addedValue) external {}

90 function afterDecreaseAllowance(address sender, address spender,
     ↪  uint256 subtractedValue) external {}
```

3.175 CVF-175

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV1.sol

Recommendation These three functions could be replaced with a single one: function beforeApprove (address sender, address spender, uint256 oldAllowance, uint256 newAllowance) external {}
Client Comment Not fixing this, because this contract is not relevant anymore v2 is the one that's in production at the moment.

Listing 175:

```
74 function beforeApprove(address sender, address spender, uint256
   ↪ amount) external {}

76 function beforeIncreaseAllowance(address msgSender, address
   ↪ spender, uint256 addedValue) external {}

78 function beforeDecreaseAllowance(address msgSender, address
   ↪ spender, uint256 subtractedValue) external {}
```

3.176 CVF-176

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LosslessControllerV1.sol

Recommendation These three functions could be replaced with a single one: function afterApprove (address sender, address spender, uint256 oldAllowance, uint256 newAllowance) external {}
Client Comment Not fixing this, because this contract is not relevant anymore v2 is the one that's in production at the moment.

Listing 176:

```
82 function afterApprove(address sender, address spender, uint256
   ↪ amount) external {}

88 function afterIncreaseAllowance(address sender, address spender,
   ↪  uint256 addedValue) external {}

90 function afterDecreaseAllowance(address sender, address spender,
   ↪  uint256 subtractedValue) external {}
```

3.177 CVF-177

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessStaking.sol

Recommendation This interface should be defined in a file named "ILssStaking.sol".

Listing 177:

```
4 interface ILssStaking {
```

3.178 CVF-178

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The return type should be "IERC20".

Listing 178:

```
5 function stakingToken() external returns(address);
```

3.179 CVF-179

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The return type should be "ILssReporting".

Listing 179:

```
6 function losslessReporting() external returns(address);
```

3.180 CVF-180

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The return type should be "ILssController".

Listing 180:

```
7 function losslessController() external returns(address);
```

3.181 CVF-181

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The return type should be "ILssGovernance".

Listing 181:

```
8 function losslessGovernance() external returns(address);
```

3.182 CVF-182

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ILosslessStaking.sol

Description This function doesn't scale.
Recommendation Consider adding two separate functions: one to obtain the number of stakers and another to obtain the staker by an index.

Listing 182:

```
10 function stakers() external returns(address[] memory);
```

3.183 CVF-183

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** ILosslessStaking.sol

Recommendation The signature of this function should be: function stakers (uint256 reportId) external returns (address[] memory);
Client Comment Not relevant anymore.

Listing 183:

```
10 function stakers() external returns(address[] memory);
```

3.184 CVF-184

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The argument type should be "ILssReporting".

Listing 184:

```
20 function setLssReporting(address _losslessReporting) external;
```

3.185 CVF-185

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The argument type should be "IERC20".

Listing 185:

```
21 function setStakingToken(address _stakingToken) external;
```

3.186 CVF-186

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The argument type should be "ILssGovernance".

Listing 186:

```
22 function setLosslessGovernance(address _losslessGovernance)
   ↪ external;
```

3.187 CVF-187

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** ILosslessStaking.sol

Recommendation Events are usually named via nouns, such as "Stake".

Listing 187:

```
28 event Staked(address indexed token, address indexed account,
   ↪ uint256 reportId);
```

3.188 CVF-188

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessStaking.sol

Recommendation The type of the "token" argument should be "ILERC20".

Listing 188:

```
28 event Staked(address indexed token, address indexed account,
   ↪ uint256 reportId);
```

3.189 CVF-189

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessStaking.sol

Recommendation The "reportId" parameter should be indexed.

Listing 189:

```
28 event Staked(address indexed token, address indexed account,
     ↪ uint256 reportId);
```

3.190 CVF-190

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessReporting.sol

Recommendation This interface should be defined in a file named "ILssReporting.sol".

Listing 190:

```
4 interface ILssReporting {
```

3.191 CVF-191

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The return type should be "ILssController".

Listing 191:

```
4 interface ILssReporting {
```

3.192 CVF-192

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The return type should be "IERC20".

Listing 192:

```
12 function stakingToken() external returns(address);
```

3.193 CVF-193

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The return type should be "ILssController".

Listing 193:

```
13 function losslessController() external returns(address);
```

3.194 CVF-194

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The return type should be "ILssGovernance".

Listing 194:

```
14 function losslessGovernance() external returns(address);
```

3.195 CVF-195

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The "token" argument should have type "ILERC20".

Listing 195:

```
23 function report(address token, address account) external returns
   ↪ (uint256);
```

3.196 CVF-196

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The argument type should be "IERC20".

Listing 196:

```
28 function setStakingToken(address _stakingToken) external;
```

3.197 CVF-197

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessReporting.sol

Recommendation The argument type should be "ILssGovernance".

Listing 197:

```
29 function setLosslessGovernance(address _losslessGovernance)
   ↪ external;
```

3.198 CVF-198

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** ILosslessReporting.sol

Recommendation Events are usually named via nouns, such as "Report", "SecondReport", or "ReportingAmountChange".

Listing 198:

```
40 event ReportSubmitted(address indexed token, address indexed
   ↪ account, uint256 reportId);
event SecondReportsubmitted(address indexed token, address
   ↪ indexed account, uint256 reportId);
event ReportingAmountChanged(uint256 indexed newAmount);
```

3.199 CVF-199

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessReporting.sol

Recommendation The "reportId" parameter should be indexed.

Listing 199:

```
40 event ReportSubmitted(address indexed token, address indexed
   ↪ account, uint256 reportId);
event SecondReportsubmitted(address indexed token, address
   ↪ indexed account, uint256 reportId);
```

3.200 CVF-200

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessGovernance.sol

Recommendation This interface should be defined in a file named "ILssGovernance.sol".

Listing 200:

```
4 interface ILssGovernance {
```

3.201 CVF-201

- **Severity** Moderate
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The return type for these functions should be "uint256".

Listing 201:

```
5 function lssTeamVoteIndex() external view returns(address);
  function tokenOwnersVoteIndex() external view returns(address);
  function committeeVoteIndex() external view returns(address);
  function committeeMembersCount() external view returns(address);
  function walletDisputePeriod() external view returns(address);
```

3.202 CVF-202

- **Severity** Moderate
- **Status** Info
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The signature for this function should be: function setStakingToken (IERC20 stakingToken) external;
Client Comment Not relevant anymore.

Listing 202:

```
10 function setStakingToken() external view returns (address);
```

3.203 CVF-203

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The return type for this function should be "IERC20".
Client Comment Not relevant anymore.

Listing 203:

```
11 function stakingToken() external view returns (address);
```

3.204 CVF-204

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The return type for this function should be "ILssStaking".

Listing 204:

```
12 function losslessStaking() external view returns (address);
```

3.205 CVF-205

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The return type for this function should be "ILssReporting".

Listing 205:

```
13 function losslessReporting() external view returns (address);
```

3.206 CVF-206

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The return type for this function should be "ILssController".

Listing 206:

```
14 function losslessController() external view returns (address);
```

3.207 CVF-207

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ILosslessGovernance.sol

Recommendation These three functions could be replaced with a single function: function vote ‘(uint256 reportId, bool vote) external;
Client Comment We’ve decided to keep the code as it is.

Listing 207:

```
25 function losslessVote(uint256 reportId, bool vote) external;
   function tokenOwnersVote(uint256 reportId, bool vote) external;
   function committeeMemberVote(uint256 reportId, bool vote)
      ↪ external;
```

3.208 CVF-208

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessGovernance.sol

Description Indexed event parameters of non-atomic types works in a non-obvious way, as only the hashes of the actual values are included into an event.
Recommendation Consider declaring the parameters as non-indexed or emitting a separate event for event added/removed committee member.

Listing 208:

```
35 event NewCommitteeMembers(address[] indexed members);
   event CommitteeMembersRemoved(address[] indexed members);
```

3.209 CVF-209

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** ILosslessGovernance.sol

Recommendation Events are usually named via nouns such as "CommitteeMembersRemoval" etc.

Listing 209:

```
36 event CommitteeMembersRemoved(address[] indexed members);
   event LosslessTeamVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event TokenOwnersVoted(uint256 indexed reportId, bool indexed
      ↪ vote);
   event CommitteeMemberVoted(uint256 indexed reportId, address
      ↪ indexed member, bool indexed vote);
40 event ReportResolved(uint256 indexed reportId, bool indexed
      ↪ resolution);
   event WalletProposed(uint256 indexed reportId, address indexed
      ↪ wallet);
   event WalletRejected(uint256 indexed reportId, address indexed
      ↪ wallet);
   event FundsRetrieved(uint256 indexed reportId, address indexed
      ↪ wallet);
   event CompensationRetrieved(address indexed wallet);
   event LosslessClaimed(address indexed token, uint256 indexed
      ↪ reportID);
```

3.210 CVF-210

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ILosslessGovernance.sol

Recommendation It would be more efficient to replace each of these events with a pair of events and removing bool parameters like this: event LosslessTeamVotedYes(uint256 indexed reportId); event LosslessTeamVotedNo(uint256 indexed reportId);

Listing 210:

```
37 event LosslessTeamVoted(uint256 indexed reportId, bool indexed
   ↪ vote);
   event TokenOwnersVoted(uint256 indexed reportId, bool indexed
   ↪ vote);
   event CommitteeMemberVoted(uint256 indexed reportId, address
   ↪ indexed member, bool indexed vote);
40 event ReportResolved(uint256 indexed reportId, bool indexed
   ↪ resolution);
```

3.211 CVF-211

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ILosslessGovernance.sol

Recommendation The "wallet" parameter is redundant as it could be derived from the preceding "WalletProposed" event.

Listing 211:

```
42 event WalletRejected(uint256 indexed reportId, address indexed
   ↪ wallet);
   event FundsRetrieved(uint256 indexed reportId, address indexed
   ↪ wallet);
```

3.212 CVF-212

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ILosslessGovernance.sol

Description These events should have an additional "amount" parameter for the actual transferred amount.

Listing 212:

```
43 event FundsRetrieved(uint256 indexed reportId, address indexed
   ↪ wallet);
event CompensationRetrieved(address indexed wallet);
event LosslessClaimed(address indexed token, uint256 indexed
   ↪ reportID);
```

3.213 CVF-213

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessGovernance.sol

Recommendation The type of the "token" parameter should be "ILERC20".

Listing 213:

```
45 event LosslessClaimed(address indexed token, uint256 indexed
   ↪ reportID);
```

3.214 CVF-214

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessERC20.sol

Recommendation This interface should be defined in a file named "ILERC20.sol".

Listing 214:

```
4 interface ILERC20 {
```

3.215 CVF-215

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessERC20.sol

Description This interface doesn't define any events.
Recommendation Consider defining at least the standard ERC-20 events: "Transfer" and "Approval". Also, consider defining Lossless-specific events.

Listing 215:

```
4 interface ILERC20 {
```

3.216 CVF-216

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ILosslessERC20.sol

Description These two functions return the same value.
Recommendation Consider removing one of them.
Client Comment The 'getAdmin()' is required for backwards compatibility.

Listing 216:

```
6 function admin() external view returns (address);

17 function getAdmin() external view returns (address);
```

3.217 CVF-217

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessControllerV3.sol

Recommendation This interface should be in a file named "ILssController.sol".

Listing 217:

```
5 interface ILssController {
```

3.218 CVF-218

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The type of the “token” arguments should be "ILERC20".

Listing 218:

```
function getLockedAmount(address token, address account)
    ↪ external view returns (uint256);
function getAvailableAmount(address token, address account)
    ↪ external view returns (uint256 amount);
function retrieveBlacklistedFunds(address[] calldata _addresses,
    ↪  address token, uint256 reportId) external returns(uint256
    ↪ );

function tokenLockTimeframe(address token) external view returns
    ↪  (uint256);
function proposedTokenLockTimeframe(address token) external view
    ↪  returns (uint256);
function changeSettlementTimelock(address token) external view
    ↪ returns (uint256);
function isNewSettlementProposed(address token) external view
    ↪ returns (bool);

function proposeNewSettlementPeriod(address token, uint256
    ↪ _seconds) external;
function executeNewSettlementPeriod(address token) external;
function activateEmergency(address token) external;
function deactivateEmergency(address token) external;

function removeProtectedAddress(address token, address
    ↪ protectedAddresss) external;

event NewSettlementPeriodProposed(address token, uint256
    ↪ _seconds);
event SettlementPeriodChanged(address token, uint256
    ↪ proposedTokenLockTimeframe);
```

3.219 CVF-219

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** ILosslessControllerV3.sol

Description Most of the argument names don't have underscore ("_") prefix, some do have.
Recommendation Consider using a consistent naming schema.

Listing 219:

```
 8 function retrieveBlacklistedFunds(address[] calldata _addresses,
     ↪  address token, uint256 reportId) external returns(uint256
     ↪ );

13 function whitelist(address _adr) external view returns (bool);

15 function getReporterPayoutStatus(address _reporter, uint256
     ↪ reportId) external view returns (bool);
   function setReporterPayoutStatus(address _reporter, bool status,
     ↪  uint256 reportId) external;
   function blacklist(address _adr) external view returns (bool);

38 function setStakingToken(address _stakingToken) external;

43 function setDexList(address[] calldata _dexList, bool value)
     ↪ external;
   function setWhitelist(address[] calldata _addrList, bool value)
     ↪ external;
   function addToBlacklist(address _adr) external;
   function resolvedNegatively(address _adr) external;
   function setStakingContractAddress(address _adr) external;
   function setReportingContractAddress(address _adr) external;
   function setGovernanceContractAddress(address _adr) external;
50 function proposeNewSettlementPeriod(address token, uint256
     ↪ _seconds) external;

68 event NewSettlementPeriodProposed(address token, uint256
     ↪ _seconds);
```

3.220 CVF-220

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ILosslessControllerV3.sol

Description It is unclear what this "erroneous compensation" is.
Recommendation Consider documenting.

Listing 220:

```
9 function erroneousCompensation() external view returns (uint256)
    ↪ ;
```

3.221 CVF-221

- **Severity** Minor
- **Status** Info
- **Category** Documentation
- **Source** ILosslessControllerV3.sol

Description The semantics of the returned values is unclear.
Recommendation Consider documenting.
Client Comment Not relevant anymore.

Listing 221:

```
15 function getReporterPayoutStatus(address _reporter, uint256
    ↪ reportId) external view returns (bool);
function setReporterPayoutStatus(address _reporter, bool status,
    ↪  uint256 reportId) external;
```

3.222 CVF-222

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The return type should be "IERC20".
Client Comment Not relevant anymore.

Listing 222:

```
22 function stakingToken() external view returns (address);
```

3.223 CVF-223

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The return type should be "ILssStaking".

Listing 223:

```
23 function losslessStaking() external view returns (address);
```

3.224 CVF-224

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The return type should be "ILssReporting".

Listing 224:

```
24 function losslessReporting() external view returns (address);
```

3.225 CVF-225

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The argument type should be "IERC20".
Client Comment Not relevant anymore.

Listing 225:

```
38 function setStakingToken(address _stakingToken) external;
```

3.226 CVF-226

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The argument type should be "ILssStaking".

Listing 226:

```
47 function setStakingContractAddress(address _adr) external;
```

3.227 CVF-227

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The argument type should be "ILssReporting".

Listing 227:

```
48 function setReportingContractAddress(address _adr) external;
```

3.228 CVF-228

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** ILosslessControllerV3.sol

Recommendation The argument type should be "ILssGovernance".

Listing 228:

```
49 function setGovernanceContractAddress(address _adr) external;
```

3.229 CVF-229

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ILosslessControllerV3.sol

Recommendation These three function could be replaced with a single one: function beforeApprove (address sender, address spender, uint256 oldAllowance, uint256 newAllowance);
Client Comment These contracts are already deployed and widely used, we cannot introduce any interface changes on LosslessController as that would brick legacy tokens.

Listing 229:

```
58 function beforeApprove(address sender, address spender, uint256
     ↪ amount) external;
   function beforeIncreaseAllowance(address msgSender, address
     ↪ spender, uint256 addedValue) external;
60 function beforeDecreaseAllowance(address msgSender, address
     ↪ spender, uint256 subtractedValue) external;
```

3.230 CVF-230

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** ILosslessControllerV3.sol

Recommendation Events are usually named via nouns such as: "AdminChange" or just "Admin", "RecoveryAdminChange" or just "RecoveryAdmin" etc.

Listing 230:

```
62 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event GuardianSet(address indexed oldGuardian, address indexed
   ↪ newGuardian);
event ProtectedAddressSet(address indexed token, address indexed
   ↪  protectedAddress, address indexed strategy);
event RemovedProtectedAddress(address indexed token, address
   ↪ indexed protectedAddress);
event NewSettlementPeriodProposed(address token, uint256
   ↪ _seconds);
event SettlementPeriodChanged(address token, uint256
   ↪ proposedTokenLockTimeframe);
```

3.231 CVF-231

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ILosslessControllerV3.sol

Recommendation The previous value parameters are redundant, as the previous value could be derived from the previous event.
Client Comment Partially fixed. We won't change events that are related to vault protection because they are already used in production.

Listing 231:

```
62 event AdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event RecoveryAdminChanged(address indexed previousAdmin,
   ↪ address indexed newAdmin);
event PauseAdminChanged(address indexed previousAdmin, address
   ↪ indexed newAdmin);
event GuardianSet(address indexed oldGuardian, address indexed
   ↪ newGuardian);
```

3.232 CVF-232

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ILosslessControllerV3.sol

Recommendation The “token” parameters should be indexed.

Listing 232:

```
68 event NewSettlementPeriodProposed(address token, uint256
   ↪ _seconds);
   event SettlementPeriodChanged(address token, uint256
   ↪ proposedTokenLockTimeframe);
```