# ABDK CONSULTING

SMART CONTRACT AUDIT

## xToken

## Terminal

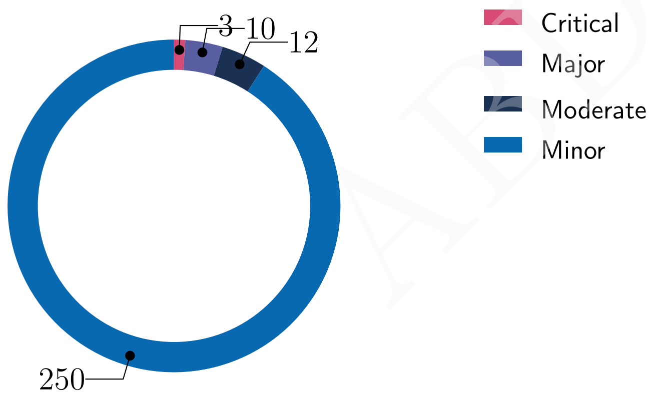
**Solidity**

abdk.consulting

# SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
10th March 2022

We've been asked to review the 24 files in a Github repository. We found 3 critical, 10 major, and a few less important issues. All critical and major issues were fixed.



## Findings

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

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

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

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

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

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

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

| ID | Severity | Category | Status |
|---|---|---|---|
| CVF-208 | Minor | Documentation | Info |
| CVF-209 | Minor | Bad datatype | Info |
| CVF-210 | Minor | Bad datatype | Info |
| CVF-211 | Minor | Bad datatype | Info |
| CVF-212 | Minor | Bad naming | Info |
| CVF-213 | Minor | Procedural | Info |
| CVF-214 | Minor | Procedural | Info |
| CVF-215 | Minor | Procedural | Info |
| CVF-216 | Moderate | Procedural | Fixed |
| CVF-217 | Minor | Bad datatype | Info |
| CVF-218 | Minor | Bad datatype | Info |
| CVF-219 | Minor | Bad datatype | Info |
| CVF-220 | Minor | Bad datatype | Info |
| CVF-221 | Minor | Documentation | Info |
| CVF-222 | Minor | Bad datatype | Info |
| CVF-223 | Minor | Bad datatype | Info |
| CVF-224 | Minor | Bad naming | Info |
| CVF-225 | Minor | Documentation | Info |
| CVF-226 | Minor | Documentation | Info |
| CVF-227 | Minor | Bad datatype | Info |
| CVF-228 | Minor | Bad datatype | Info |
| CVF-229 | Minor | Procedural | Info |
| CVF-230 | Minor | Bad datatype | Info |
| CVF-231 | Minor | Documentation | Info |
| CVF-232 | Minor | Bad datatype | Info |
| CVF-233 | Minor | Procedural | Info |
| CVF-234 | Minor | Bad datatype | Info |
| CVF-235 | Minor | Bad datatype | Info |
| CVF-236 | Minor | Bad datatype | Info |
| CVF-237 | Minor | Bad datatype | Info |

| ID | Severity | Category | Status |
|---|---|---|---|
| CVF-238 | Minor | Bad datatype | Info |
| CVF-239 | Minor | Bad datatype | Info |
| CVF-240 | Minor | Bad datatype | Info |
| CVF-241 | Minor | Bad datatype | Info |
| CVF-242 | Minor | Bad datatype | Info |
| CVF-243 | Minor | Procedural | Info |
| CVF-244 | Minor | Bad datatype | Info |
| CVF-245 | Minor | Bad datatype | Info |
| CVF-246 | Minor | Procedural | Info |
| CVF-247 | Minor | Documentation | Info |
| CVF-248 | Minor | Procedural | Info |
| CVF-249 | Minor | Bad datatype | Info |
| CVF-250 | Minor | Documentation | Info |
| CVF-251 | Moderate | Procedural | Fixed |
| CVF-252 | Minor | Bad datatype | Info |
| CVF-253 | Minor | Documentation | Info |
| CVF-254 | Minor | Documentation | Info |
| CVF-255 | Minor | Documentation | Info |
| CVF-256 | Minor | Documentation | Info |
| CVF-257 | Minor | Bad naming | Info |
| CVF-258 | Minor | Documentation | Info |
| CVF-259 | Minor | Documentation | Info |
| CVF-260 | Minor | Bad datatype | Info |
| CVF-261 | Minor | Bad datatype | Info |
| CVF-262 | Minor | Bad datatype | Info |
| CVF-263 | Minor | Procedural | Info |
| CVF-264 | Minor | Bad naming | Info |
| CVF-265 | Minor | Documentation | Info |
| CVF-266 | Minor | Documentation | Fixed |
| CVF-267 | Minor | Suboptimal | Info |

| ID | Severity | Category | Status |
|---|---|---|---|
| CVF-268 | Minor | Suboptimal | Info |
| CVF-269 | Minor | Documentation | Info |
| CVF-270 | Minor | Unclear behavior | Info |
| CVF-271 | Minor | Bad datatype | Info |
| CVF-272 | Minor | Bad datatype | Info |
| CVF-273 | Minor | Bad datatype | Info |
| CVF-274 | Minor | Bad datatype | Info |
| CVF-275 | Minor | Bad datatype | Info |

## Contents

# 1 Document properties

## Version

| Version | Date | Author | Description |
|---|---|---|---|
| 0.1 | February 23, 2022 | D. Khovratovich | Initial Draft |
| 0.2 | February 23, 2022 | D. Khovratovich | Minor revision |
| 1.0 | February 23, 2022 | D. Khovratovich | Release |
| 1.1 | March 10, 2022 | D. Khovratovich | Client comments and fixes are added |
| 2.0 | March 10, 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 90e6ed9 commit files:

- interfaces/ICLR.sol
- interfaces/ICLRDeployer.sol
- interfaces/IERC20.sol
- interfaces/IERC20Extended.sol
- interfaces/ILMTerminal.sol
- interfaces/IProxyAdmin.sol
- interfaces/IRewardEscrow.sol
- interfaces/IStakedCLRToken.sol
- interfaces/IStakingRewards.sol
- interfaces/IxTokenManager.sol
- libraries/UniswapLibrary.sol
- libraries/Utils.sol
- proxies/CLRProxy.sol
- proxies/LMTerminalProxy.sol
- proxies/ProxyAdmin.sol
- proxies/StakedCLRTokenProxy.sol
- staking/proxies/StakingRewardsProxy.sol
- staking/RewardEscrow.sol
- staking/StakingRewards.sol
- BlockLock.sol
- CLR.sol
- CLRDeployer.sol
- LMTerminal.sol
- StakedCLRToken.sol

The fixes were provided in a new 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** Info
- **Category** Procedural
- **Source** StakedCLRToken.sol

**Recommendation** Should be "^0.7.0" unless there is something special about this particular version. Also relevant for the next files: CLR.sol, UniswapLibrary.sol, LMTerminal.sol, CLRDeployer.sol, RewardEscrow.sol, BlockLock.sol, StakingRewards.sol, StakingRewardsProxy.sol, ProxyAdmin.sol, StakedCLRTokenProxy.sol, CLRProxy.sol, LMTerminalProxy.sol, Utils.sol, IxTokenManager.sol, IStakedCLRToken.sol, IRewardEscrow.sol, IStakingRewards.sol, IProxyAdmin.sol, ILMTerminal.sol, IERC20Extended.sol, ICLRDeployer.sol, IERC20.sol, ICLR.sol.
**Client Comment** 0.7.6 is the version with which the Uniswap V3 contracts are compiled with, so there is no reason to compile the rest of the code with a different solidity version.

Listing 1:

```
2 pragma solidity 0.7.6;
```

## 3.2 CVF-2

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

**Recommendation** The type of this variable should be "ICLR".
**Client Comment** Why should the type be ICLR ? I'm using this variable only in the onlyCLRPool modifier, which checks if the msg.sender is the pool address. If it's ICLR, I'll need to convert it to address type on every check.

Listing 2:

```
13 address public clrPool;
```

## 3.3 CVF-3

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

**Description** This check is redundant. It is anyway possible to pass a dead CLR pool address.
**Recommendation** Consider removing this check.
**Client Comment** Decided to leave it as it is.

Listing 3:

```
22 require(_clrPool != address(0), "CLR Pool cannot be 0x0 address
    ↪ ");
```

### 3.4 CVF-4

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

**Description** These functions always return true.
**Recommendation** Consider returning nothing.
**Client Comment** Decided to leave it as it is.

Listing 4:

```
return true;
```

```
return true;
```

### 3.5 CVF-5

- **Severity** Major
- **Category** Bad naming
- **Status** Fixed
- **Source** StakedCLRToken.sol

**Description** These functions use the "notLocked" modifier but don't call the "lock" function.
**Recommendation** Consider calling the "lock" function at the beginning of the functions.

Listing 5:

```
function transfer(address recipient, uint256 amount)
```

```
function transferFrom(
```

### 3.6 CVF-6

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

**Description** This import is not used.
**Recommendation** Consider removing it.
**Client Comment** Removed.

Listing 6:

```
import "./BlockLock.sol";
```

## 3.7 CVF-7

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

**Description** This contract doesn't implement the "ICLR" interface, which is error prone, as compiler cannot check that all the interface functions are actually implemented.
**Client Comment** Fixed by implementing all interface functions manually. Cannot inherit interface due to function clashes with StakingRewards.sol

Listing 7:

```
20  contract CLR is
```

## 3.8 CVF-8

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

**Description** The number format of these value is unclear.
**Recommendation** Consider documenting.
**Client Comment** Seems pretty straightforward to me, we use divisors instead of percentage multipliers.

Listing 8:

```
31  uint256 private constant SWAP_SLIPPAGE = 50; // 2%
    uint256 private constant MINT_BURN_SLIPPAGE = 100; // 1%
```

## 3.9 CVF-9

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

**Description** The same constants are defined in ‘UniswapLibrary‘ contract.
**Recommendation** Consider declaring them in one file and then import.
**Client Comment** Decided to leave it as it is.

Listing 9:

```
31  uint256 private constant SWAP_SLIPPAGE = 50; // 2%
    uint256 private constant MINT_BURN_SLIPPAGE = 100; // 1%
```

## 3.10 CVF-10

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

**Description** There are no access level specified for these variables, so internal access will be used by default.
**Recommendation** Consider explicitly specifying an access level.
**Client Comment** Decided to leave it as it is.

Listing 10:

```
int24 tickLower;
int24 tickUpper;

uint160 priceLower;
uint160 priceUpper;

uint32 twapPeriod; // Time period of twap

address terminal;
```

## 3.11 CVF-11

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

**Description** The number format for these variables is unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is.

Listing 11:

```
uint256 public tradeFee; // xToken Trade Fee as a divisor (100 =
    ↪ 1%)
uint24 public poolFee;
```

## 3.12 CVF-12

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

**Recommendation** The type of this variable should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 12:

```
address public uniswapPool;
```

## 3.13 CVF-13

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

**Recommendation** The type of this field should be "ISwapRouter".
**Client Comment** Decided to leave it as address.

Listing 13:

```
address router;
```

## 3.14 CVF-14

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

**Recommendation** The type of this field should be "IQuoter".
**Client Comment** Decided to leave it as address.

Listing 14:

```
address quoter;
```

## 3.15 CVF-15

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

**Recommendation** The type of this field should be "INonfungiblePositionManager".
**Client Comment** Decided to leave it as address.

Listing 15:

```
address positionManager;
```

## 3.16 CVF-16

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

**Recommendation** The type of this field should be "IERC20 []".
**Client Comment** Decided to leave it as address[].

Listing 16:

```
address[] rewardTokens;
```

## 3.17 CVF-17

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

**Recommendation** The type of this field should be "IRewardEscrow".
**Client Comment** Decided to leave it as address.

Listing 17:

```
address rewardEscrow;
```

## 3.18 CVF-18

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

**Recommendation** Events are usually named via nouns, such as "FeeCollection", "Manager", "Withdrawal".
**Client Comment** Decided to leave them as they are.

Listing 18:

```
78 event FeeCollected(uint256 token0Fee, uint256 token1Fee);
   event ManagerSet(address indexed manager);

81 event Withdraw(address indexed user, uint256 amount0, uint256
     ↪ amount1);
```

## 3.19 CVF-19

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

**Description** There are no range checks for these arguments.
**Recommendation** Consider adding appropriate checks.
**Client Comment** Decided to leave it as it is.

Listing 19:

```
85 int24 _tickLower,
   int24 _tickUpper,
   uint256 _tradeFee,
```

### 3.20 CVF-20

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

**Description** The number format of this argument is unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is.

Listing 20:

```
uint256 _tradeFee,
```

### 3.21 CVF-21

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

**Recommendation** The type of these arguments should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 21:

```
    address _token0,
    address _token1,

function withdrawToken(address token, address receiver)

function initializeReward(uint256 rewardAmount, address token)
```

### 3.22 CVF-22

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

**Description** These arguments are redundant, as their values could be derived from "_uniswap-Pool".
**Client Comment** Decided to leave it as it is.

Listing 22:

```
address _token0,
address _token1,
```

## 3.23 CVF-23

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

**Recommendation** The type of this argument should be "IStakedCLRToken".
**Client Comment** Decided to leave it as address.

Listing 23:

```
address _stakedToken,
```

## 3.24 CVF-24

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

**Recommendation** The type of this argument should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 24:

```
address _uniswapPool,
```

## 3.25 CVF-25

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

**Description** The "decimals" property in ERC-20 is used by UI to render token amount in a human-readable way. Using this property in smart contracts is discourages.
**Recommendation** Consider treating token amounts as integers.
**Client Comment** Decided to leave it as it is.

Listing 25:

```
    token0Decimals = IERC20Extended(_token0).decimals();
    token1Decimals = IERC20Extended(_token1).decimals();

function getToken0AmountInWei(uint256 amount)

function getToken1AmountInWei(uint256 amount)
```

## 3.26 CVF-26

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

**Description** This will revert in case a token has more than 18 decimals.
**Client Comment** Tokens with more than 18 decimals will not be supported for Terminal V1.

Listing 26:

```
10**(TOKEN_DECIMAL_REPRESENTATION.sub(token0Decimals));

10**(TOKEN_DECIMAL_REPRESENTATION.sub(token1Decimals));
```

## 3.27 CVF-27

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

**Recommendation** These values should be named constants.
**Client Comment** Decided to leave it as it is.

Listing 27:

```
poolFee = 3000;

twapPeriod = 3600;
```

## 3.28 CVF-28

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

**Description** In the former case "token1" is encoded as value 1, while in the latter cases as any non-zero value.
**Recommendation** Consider interpreting input asset codes in a consistent way.
**Client Comment** Decided to leave it as it is.

Listing 28:

```
*  @param inputAsset asset to mint with (0 - token 0, 1 - token
  ↪ 1)

* @param inputAsset - use token0 if 0, token1 else
```

## 3.29 CVF-29

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

**Recommendation** The type of the "inputAsset" arguments should be a enum with two valid values, or even bool.
**Client Comment** Decided to leave it as it is.

Listing 29:

```
function deposit(uint8 inputAsset, uint256 amount) external
    ↪ whenNotPaused {

function calculateAmountsMintedSingleToken(uint8 inputAsset,
    ↪ uint256 amount)
```

## 3.30 CVF-30

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

**Recommendation** This function should return the amount of receipt tokens minted.
**Client Comment** Decided to leave it as it is.

Listing 30:

```
function deposit(uint8 inputAsset, uint256 amount) external
    ↪ whenNotPaused {
```

## 3.31 CVF-31

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

**Description** The expression "amount0 > token0Balance" is potentially calculated twice.
**Recommendation** Consider calculating once and reusing.
**Client Comment** Decided to leave it as it is.

Listing 31:

```
if (amount0 > token0Balance || amount1 > token1Balance) {
    amount0 = amount0 > token0Balance ? token0Balance : amount0;
```

## 3.32 CVF-32

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

**Description** The expression "amount1 > token1Balance" is potentially calculated twice.
**Recommendation** Consider calculating once and reusing.
**Client Comment** Decided to leave it as it is.

Listing 32:

```
163 if (amount0 > token0Balance || amount1 > token1Balance) {

165     amount1 = amount1 > token1Balance ? token1Balance : amount1;
```

## 3.33 CVF-33

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

**Description** These assignments are redundant, as the assigned values are overwritten in the next line.
**Recommendation** Consider passing ternary expressions directly to the "calculatePoolMintedAmounts" function.
**Client Comment** Decided to leave it as it is.

Listing 33:

```
164 amount0 = amount0 > token0Balance ? token0Balance : amount0;
    amount1 = amount1 > token1Balance ? token1Balance : amount1;
```

## 3.34 CVF-34

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

**Recommendation** This function should return the amounts of obtained tokens.
**Client Comment** Decided to leave it as it is.

Listing 34:

```
189 function withdraw(uint256 amount) public {
```

## 3.35 CVF-35

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

**Description** Due to these additional, more liquidity would be burned than needed, which means that less liquidity will remain in the pool for the remaining stakers. In case all the remaining stakes are withdrawn (amount == totalSupply), this will lead to an attempt to burn more liquidity, than available, thus the contract will revert.
**Client Comment** Duplicate of CVF-36. Fixed in Major issues PR.

Listing 35:

```
212 uint256 unstakeAmount0 = amount0.add(amount0.div(
    ↪ MINT_BURN_SLIPPAGE));
uint256 unstakeAmount1 = amount1.add(amount1.div(
    ↪ MINT_BURN_SLIPPAGE));
```

## 3.36 CVF-36

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

**Description** While "amount0" and "amount1" values are in balance with the current token distribution in the pool, and withdrawing these amounts will not move the price, after adding slippage margins, the amount could not be in balance anymore.
**Recommendation** Consider adding the slippage margin to the original liquidity amount before calling the "getAmountForLiquidity" function.

Listing 36:

```
212 uint256 unstakeAmount0 = amount0.add(amount0.div(
    ↪ MINT_BURN_SLIPPAGE));
uint256 unstakeAmount1 = amount1.add(amount1.div(
    ↪ MINT_BURN_SLIPPAGE));
```

## 3.37 CVF-37

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

**Description** Due to these "subzero" operations, the amounts returns by the "getBufferBalance" and "getStakedBalance" could be higher than the actual values in case of balance is negative, but the other balance is positive.
**Recommendation** Consider returning singled values here.
**Client Comment** Decided to leave it as it is.

Listing 37:

```
316 UniswapLibrary.subZero(

330 UniswapLibrary.subZero(
```

## 3.38 CVF-38

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

**Description** This function assumes that underlying tokens were already transferred to the contract.
**Recommendation** Consider mentioning this fact in the documentation comment.
**Client Comment** Decided to leave it as it is.

Listing 38:

```
366 function calculateMintAmount(uint256 _amount, uint256
    ↪ totalSupply)
```

## 3.39 CVF-39

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

**Description** The meaning of this argument is unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is.

Listing 39:

```
409 address sender
```

## 3.40 CVF-40

- **Severity** Minor
- **Category** Flaw
- **Status** Info
- **Source** CLR.sol

**Description** It is not explicitly checked that "tokenId" is not zero, i.e. that position was already created.
**Recommendation** Consider adding such explicit checks.
**Client Comment** Decided to leave it as it is.

Listing 40:

```
454 tokenId: tokenId,

507 tokenId

581     tokenId: tokenId,

666 tokenId: tokenId,

703 tokenId: tokenId,

736 tokenId,

746 tokenId
```

## 3.41 CVF-41

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

**Description** This value is guaranteed to be zero here.
**Recommendation** Consider passing zero explicitly.

Listing 41:

```
554 tokenId: tokenId,
```

## 3.42 CVF-42

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

**Description** This function allows rescuing ERC-20 tokens only, but not other classes of assets.
**Recommendation** Consider supporting all asset classes, but allowing the admin to initiate calls of arbitrary functions on arbitrary contracts, except for this contract and contracts this contract knows, such as token0, token1, stakedToken, Uniswap pool, position manger etc.
**Client Comment** Function was removed.

Listing 42:

```
* Emergency function in case of errant transfer
* of any token directly to contract
```

## 3.43 CVF-43

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

**Description** Despite the name and comment, there could be at most one manager and this functions sets the manager rather than adds another manager.
**Recommendation** Consider renaming.
**Client Comment** Decided to leave it as it is.

Listing 43:

```
 * @notice Add manager to CLR instance
 * @notice Managers have the same management permissions as owners

function addManager(address _manager) external onlyOwner {
```

## 3.44 CVF-44

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

**Description** These functions always return true.
**Recommendation** Consider returning nothing.
**Client Comment** Decided to leave it as it is.

Listing 44:

```
621 function pauseContract() external onlyOwnerOrManager returns (
    ↪ bool) {

626 function unpauseContract() external onlyOwnerOrManager returns (
    ↪ bool) {
```

## 3.45 CVF-45

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

**Recommendation** Consider renaming to "maxAmountIn" for readability.
**Client Comment** Decided to leave it as it is.

Listing 45:

```
654 * @param amountIn – amount as maximum input for swap, in token 0
    ↪  terms

691 * @param amountIn – amount as maximum input for swap, in token 1
    ↪  terms
```

## 3.46 CVF-46

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

**Description** The f\number format of the returned value is unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is.

Listing 46:

```
755 function getAsset0Price() public view returns (int128) {

771 function getAsset1Price() public view returns (int128) {
```

## 3.47 CVF-47

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

**Description** This function wouldn't be necessary if the "tickLower" and "tickUpper" variables would be public.
**Client Comment** Decided to leave it as it is.

Listing 47:

```
function getTicks() external view returns (int24 tick0, int24
    ↪ tick1) {
```

## 3.48 CVF-48

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

**Description** There is another "IERC20" interface in "../interfaces/IERC20.sol" used in other files.
**Recommendation** Consider using the same "IERC20" interface across the code, or giving a different name to that interface. Using several interfaces with the same name makes code harder to read.
**Client Comment** Decided to leave it as it is.

Listing 48:

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

## 3.49 CVF-49

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

**Recommendation** The type of these fields should be "IERC20".
**Client Comment** Decided to leave them as address.

Listing 49:

```
address token0;
address token1;
```

## 3.50 CVF-50

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

**Description** The "decimals" property of a ERC-20 token is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.
**Recommendation** Consider treating all token amounts are integers.
**Client Comment** Decided to leave it as it is.

Listing 50:

```
35 uint256 token0DecimalMultiplier;
   uint256 token1DecimalMultiplier;
   uint256 tokenDiffDecimalMultiplier;
   uint8 token0Decimals;
   uint8 token1Decimals;
```

## 3.51 CVF-51

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

**Description** The number formats of these fields are unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is.

Listing 51:

```
46 uint24 poolFee;

48 uint160 priceLower;
   uint160 priceUpper;
```

## 3.52 CVF-52

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

**Recommendation** The type of this field should be "INonfungiblePositionManager".
**Client Comment** Decided to leave it as address.

Listing 52:

```
51 address positionManager;
```

### 3.53 CVF-53

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

**Recommendation** The type of this field should be "ISwapRouter".
**Client Comment** Decided to leave it as address.

Listing 53:

```
address router;
```

### 3.54 CVF-54

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

**Recommendation** The type of this field should be "IQuoter".
**Client Comment** Decided to leave it as address.

Listing 54:

```
address quoter;
```

### 3.55 CVF-55

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

**Recommendation** The type of this field should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 55:

```
address pool;
```

## 3.56 CVF-56

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

**Description** This structure looks identical to a structure defined in "Utils.sol".
**Recommendation** Consider defining this structure in a single place.
**Client Comment** Decided to leave it as it is.

Listing 56:

```
struct AmountsMinted {
    uint256 amount0ToMint;
    uint256 amount1ToMint;
    uint256 amount0Minted;
    uint256 amount1Minted;
}
```

## 3.57 CVF-57

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

**Recommendation** The type of the "_pool" and "pool" arguments should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 57:

```
function getPoolPrice(address _pool) public view returns (
    ↪ uint160) {

function getPoolPriceWithDecimals(address _pool)

function getPoolLiquidity(address _pool) public view returns (
    ↪ uint128) {

    address pool

    address pool

    address pool

    address pool,

    address pool,

    address pool,

    address pool,
```

## 3.58 CVF-58

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

**Description** The "sqrtRatioX96" variable is redundant.
**Recommendation** Just give a name to the returned value and use it instead.
**Client Comment** Decided to leave it as it is.

Listing 58:

```
(uint160 sqrtRatioX96, , , , , , ) = pool.slot0();
```

## 3.59 CVF-59

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** UniswapLibrary.sol

**Description** Phantom overflow is possible here, i.e. a situation when the final calculation result would fit into the destination type, but some intermediary calculations overflow.
**Recommendation** Consider using a non-overflowing method, such as the "muldiv" function: https://xn--2-umb.com/21/muldiv/index.html , or some tricks described here: https://medium.com/coinmonks/math-in-solidity-part-3-percents-and-proportions-4db014e080b1 . Alternatively, first multiply the "sqrtRatioX96" value by 1e6, shift right by 96 bits, and then square. This way overflow will never be possible.

Listing 59:

```
87 uint256(sqrtRatioX96).mul(uint256(sqrtRatioX96)).mul(1e12) >>
     ↪ 192;
```

## 3.60 CVF-60

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

**Description** This variable is redundant, as its value is used only once.
**Recommendation** Consider using the expression instead.
**Client Comment** Decided to leave it as it is for readability.

Listing 60:

```
94 IUniswapV3Pool pool = IUniswapV3Pool(_pool);
```

## 3.61 CVF-61

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

**Description** The “decimals” property of a ERC-20 token is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.
**Recommendation** Consider treating token amounts as integers.
**Client Comment** Decided to leave it as it is.

Listing 61:

```
if (token1Decimals > token0Decimals) {

} else if (token0Decimals > token1Decimals) {

amountIn = getToken0AmountInNativeDecimals(

amountOut = getToken1AmountInNativeDecimals(

amountIn = getToken1AmountInNativeDecimals(

amountOut = getToken0AmountInNativeDecimals(

token0Balance = getToken0AmountInNativeDecimals(

token1Balance = getToken1AmountInNativeDecimals(

token0Balance = getToken0AmountInNativeDecimals(

token1Balance = getToken1AmountInNativeDecimals(

        getToken0AmountInNativeDecimals(

        getToken0AmountInNativeDecimals(

            amount0ToMint: getToken0AmountInWei(

            amount1ToMint: getToken1AmountInWei(

            amount0Minted: getToken0AmountInWei(

            amount1Minted: getToken1AmountInWei(

amount0 = getToken0AmountInWei(

amount1 = getToken1AmountInWei(
```

## 3.62 CVF-62

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

**Description** This is basically equivalent to: twap = int128 (uint256 (twap) / tokenDiffDecimalsMultiplier);
**Client Comment** Removed function.

Listing 62:

```
twap = ABDKMath64x64.mul(
      twap,
      ABDKMath64x64.divu(1, tokenDiffDecimalMultiplier)
```

## 3.63 CVF-63

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

**Description** This is equivalent to: twap = toInt128 (uint256 (twap).mul (tokenDiffDecimalsMultiplier));
**Client Comment** Removed function.

Listing 63:

```
int128 multiplierFixed = ABDKMath64x64.fromUInt(
      tokenDiffDecimalMultiplier
);
twap = ABDKMath64x64.mul(twap, multiplierFixed);
```

## 3.64 CVF-64

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

**Description** These functions are very similar.
**Recommendation** Consider extracting common parts into utility functions to avoid code duplication.

Listing 64:

```
function swapToken0ForToken1(

function swapToken1ForToken0(
```

### 3.65 CVF-65

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** UniswapLibrary.sol

**Description** Phantom overflow is possible here.
**Recommendation** Consider using a non-overflowing "muldiv" function.

Listing 65:

```
amountOut = amountOut.mul(midPrice).div(1e12);
```

```
amountOut = amountOut.mul(1e12).div(midPrice);
```

### 3.66 CVF-66

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

**Description** This may reduce the output amount specified by the caller, effectively making the provided output amount to be the maximum allowed output. This doesn't make sense from economical point of view. Usually, either the maximum input or the minimum output amount is specified, but not maximum output.
**Client Comment** The reason why I've used quoteExactInputSingle before doing the swap is that in some cases the swapAmount returned by Utils.calculateSwapAmount, which is passed to swapToken0ForToken1 function as amountOut and is then converted to token 1 terms by multiplying by the price, exceeded the passed amountInMaximum, due to the price changing in the case of large swaps or in pools with little liquidity. So, the way I circumvented that is by calculate the expected amountOut of the swap.

Listing 66:

```
if (amountOutExpected < amountOut) {
    amountOut = amountOutExpected;
}
```

```
if (amountOutExpected < amountOut) {
    amountOut = amountOutExpected;
}
```

## 3.67 CVF-67

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

**Recommendation** The type of the "positionManager" arguments should be "INonfungible-PositionManager".
**Client Comment** Decided to leave it as address.

Listing 67:

```
function getPositionLiquidity(address positionManager, uint256 tokenId)

    address positionManager,

    address positionManager

    address positionManager,
```

## 3.68 CVF-68

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

**Recommendation** The maximum slippage should be passed as an argument. Hardcoding it makes the library less flexible.
**Client Comment** Decided to leave it as it is, because the argument won't be changed at all.

Listing 68:

```
            amount0Min: amount0.sub(amount0.div(MINT_BURN_SLIPPAGE)),
            amount1Min: amount1.sub(amount1.div(MINT_BURN_SLIPPAGE)),

        amount0Min: _amount0.sub(_amount0.div(MINT_BURN_SLIPPAGE)),
        amount1Min: _amount1.sub(_amount1.div(MINT_BURN_SLIPPAGE)),

        amount0Min: amount0.sub(amount0.div(MINT_BURN_SLIPPAGE)),
        amount1Min: amount1.sub(amount1.div(MINT_BURN_SLIPPAGE)),

amount0ToMint.sub(amount0ToMint.div(MINT_BURN_SLIPPAGE)) ||

amount1ToMint.sub(amount1ToMint.div(MINT_BURN_SLIPPAGE))
```

## 3.69 CVF-69

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

**Recommendation** Using a multiplier rather than a divider to specify the maximum allowed slippage would be more conventional.
**Client Comment** Decided to leave it as it is.

Listing 69:

```
            amount0Min: amount0.sub(amount0.div(MINT_BURN_SLIPPAGE)),
            amount1Min: amount1.sub(amount1.div(MINT_BURN_SLIPPAGE)),

        amount0Min: _amount0.sub(_amount0.div(MINT_BURN_SLIPPAGE)),
        amount1Min: _amount1.sub(_amount1.div(MINT_BURN_SLIPPAGE)),

        amount0Min: amount0.sub(amount0.div(MINT_BURN_SLIPPAGE)),
        amount1Min: amount1.sub(amount1.div(MINT_BURN_SLIPPAGE)),

amount0ToMint.sub(amount0ToMint.div(MINT_BURN_SLIPPAGE)) ||

amount1ToMint.sub(amount1ToMint.div(MINT_BURN_SLIPPAGE))
```

## 3.70 CVF-70

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

**Recommendation** This check could be optimized as: require (amount0 | amount1 != 0);

Listing 70:

```
require(amount0 != 0 || amount1 != 0, "Rebalance amounts are 0");
```

## 3.71 CVF-71

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

**Description** The "divuu" function is private in the original ABDKMath64x64 library and is not supposed to be used from the outside.
**Recommendation** Consider using the "divu" function instead.
**Client Comment** Decided to leave it as it is.

Listing 71:

```
667 : int128(ABDKMath64x64.divuu(mintLiquidity, poolLiquidity));
```

## 3.72 CVF-72

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** UniswapLibrary.sol

**Description** Overflow is possible when converting to "int128".
**Recommendation** Consider using the "divu" function to avoid unsafe conversion.
**Client Comment** Decided to leave it as it is.

Listing 72:

```
667 : int128(ABDKMath64x64.divuu(mintLiquidity, poolLiquidity));
```

## 3.73 CVF-73

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

**Description** This conditions was already checked inside the "calcualteSwapAmount" function.
**Recommendation** Consider taking it from there rather than calculating here again.
**Client Comment** Decided to leave it as it is.

Listing 73:

```
746 if (mul1 > mul2) {
```

## 3.74 CVF-74

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

**Recommendation** Consider adding an empty ‘else‘ clause with the comment that it corresponds to the case ‘swapamount==0‘ – for readability
**Client Comment** Decided to leave it as it is.

Listing 74:

```
}
```

## 3.75 CVF-75

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

**Recommendation** These two functions do exactly the same, consider removing one of them.
**Client Comment** Decided to leave it as it is.

Listing 75:

```
824 function getBufferToken0Balance(

841 function getBufferToken1Balance(
```

## 3.76 CVF-76

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

**Description** These two functions do exactly the same.
**Recommendation** Consider removing one of them.
**Client Comment** Decided to leave it as it is.

Listing 76:

```
861 function getToken0AmountInNativeDecimals(

875 function getToken1AmountInNativeDecimals(
```

## 3.77 CVF-77

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

**Description** The cases when a token has more than 18 decimals are not handled properly.
**Recommendation** Consider handling such cases as well or adding explicit checks to completely forbid such tokens.
**Client Comment** Tokens with more than 18 decimals will not be supported for Terminal V1.

Listing 77:

```
866 if (token0Decimals < TOKEN_DECIMAL_REPRESENTATION) {

880 if (token1Decimals < TOKEN_DECIMAL_REPRESENTATION) {

894 if (token0Decimals < TOKEN_DECIMAL_REPRESENTATION) {

908 if (token1Decimals < TOKEN_DECIMAL_REPRESENTATION) {
```

## 3.78 CVF-78

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

**Description** These two functions do exactly the same.
**Recommendation** Consider removing one of them.
**Client Comment** Decided to leave it as it is.

Listing 78:

```
889 function getToken0AmountInWei(

903 function getToken1AmountInWei(
```

## 3.79 CVF-79

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

**Description** These functions are thin wrappers for functions from “TickMath” library.
**Recommendation** Consider removing these functions and using the “TickMath” functions instead.
**Client Comment** Decided to leave it as it is.

Listing 79:

```
function getSqrtRatio(int24 tick) public pure returns (uint160)
    ↪ {

function getTickFromPrice(uint160 price) public pure returns (
    ↪ int24) {
```

## 3.80 CVF-80

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

**Recommendation** These low-level functions should be moved to some utility library.
**Client Comment** Decided to leave it as it is.

Listing 80:

```
function subAbs(uint256 amount0, uint256 amount1)

function subZero(uint256 amount0, uint256 amount1)
```

## 3.81 CVF-81

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

**Recommendation** The key type should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 81:

```
mapping(address => uint256) public rewardFeesTotal; // total
    ↪ reward fees for each reward token
```

## 3.82 CVF-82

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

**Recommendation** The type or this variable should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 82:

```
44 address public proxyAdmin; // Proxy Admin of CLR instances
```

## 3.83 CVF-83

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

**Recommendation** The type of this field should be "IERC20[]".
**Client Comment** Decided to leave it as address[].

Listing 83:

```
58 address[] rewardTokens;
```

## 3.84 CVF-84

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

**Recommendation** The type of these fields should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 84:

```
65 address token0;
   address token1;
```

## 3.85 CVF-85

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

**Recommendation** Events are usually named via nouns, such as "UniV3Pool" or "NewUniV3Pool", "IncentivizedPool" or "NewIncentivizedPool" etc.
**Client Comment** Decided to leave the names as they are.

Listing 85:

```
event DeployedUniV3Pool(

event DeployedIncentivizedPool(

event InitiatedRewardsProgram(

event ClaimFeeWithdraw(

event TokenFeeWithdraw(address indexed token, uint256 amount);
event EthFeeWithdraw(uint256 amount);
```

## 3.86 CVF-86

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

**Recommendation** The type of these parameters should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 86:

```
address indexed pool,

address indexed pool,
```

## 3.87 CVF-87

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

**Recommendation** The type of the token parameters should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 87:

```
    address indexed token0,
    address indexed token1,

    address indexed token0,
    address indexed token1,

event TokenFeeWithdraw(address indexed token, uint256 amount);
```

## 3.88 CVF-88

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

**Recommendation** The type of these parameters should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 88:

```
address indexed clrInstance,

address indexed clrInstance,
```

## 3.89 CVF-89

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

**Recommendation** The type of this parameter should be "IERC20[]".
**Client Comment** Decided to leave it as address.

Listing 89:

```
address[] rewardTokens,
```

## 3.90 CVF-90

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

**Recommendation** The type of this argument should be "IxTokenManager".
**Client Comment** Decided to leave it as address.

Listing 90:

```
address _xTokenManager,
```

## 3.91 CVF-91

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

**Recommendation** The type of this argument should be "IRewardEscrow".
**Client Comment** Decided to leave it as address.

Listing 91:

```
address _rewardEscrow,
```

## 3.92 CVF-92

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

**Recommendation** The type of thsi argument should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 92:

```
address _proxyAdmin,
```

## 3.93 CVF-93

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

**Recommendation** The type of this argument should be "ICLRDeployer".
**Client Comment** Decided to leave it as address.

Listing 93:

```
address _clrDeployer,
```

## 3.94 CVF-94

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

**Recommendation** The type of this argument should be "IUniswapV3Factory".
**Client Comment** Decided to leave it as address.

Listing 94:

```
address _uniswapFactory,
```

## 3.95 CVF-95

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

**Description** There are no range checks for these arguments.
**Recommendation** Consider adding appropriate checks.

Listing 95:

```
uint256 _deploymentFee,
uint256 _rewardFee,
uint256 _tradeFee
```

## 3.96 CVF-96

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

**Recommendation** The type for these arguments should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 96:

```
address token0,
address token1,
```

### 3.97 CVF-97

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

**Recommendation** Solidity supports multiple assignment like this: (token0, token1) = (token1, token0);
**Client Comment** Fixed in minor issues PR.

Listing 97:

```
144 address tmp = token0;
    token0 = token1;
    token1 = tmp;

193 address tmp = pool.token0;
    pool.token0 = pool.token1;
    pool.token1 = tmp;
```

### 3.98 CVF-98

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

**Description** The name is the same for all staked CLR tokens deployed. This could be confusing.
**Recommendation** Consider adding the symbol to the name.
**Client Comment** Decided to leave it as it is.

Listing 98:

```
185 "StakedCLRToken",
```

### 3.99 CVF-99

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

**Description** The semantic of this value is unclear.
**Recommendation** Consider adding a comment with the argument name.
**Client Comment** Decided to leave it as it is.

Listing 99:

```
188 false
```

## 3.100 CVF-100

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

**Recommendation** This could be simplified as: bool rewardsAreExcrowed = rewardsProgram.vestingPeriod > 0;
**Client Comment** Decided to leave it as it is.

Listing 100:

```
bool rewardsAreEscrowed = rewardsProgram.vestingPeriod > 0
    ? true
    : false;
```

## 3.101 CVF-101

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

**Description** A list of the reward tokens is potentially obtained twice: once in this function and another time inside the "_initiateRewardsProgram" function.
**Recommendation** Consider refactoring the code to avoid querying the same information several times.

Listing 101:

```
    address[] memory rewardTokens = clrPool.getRewardTokens();

_initiateRewardsProgram(clrPool, totalRewardAmounts);
```

## 3.102 CVF-102

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

**Description** Overflow is possible here.
**Recommendation** Consider using a safe add operation.
**Client Comment** Fixed (function was refactored).

Listing 102:

```
rewardFeesTotal[rewardToken] += rewardAmountFee;
```

### 3.103 CVF-103

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

**Recommendation** The type for these arguments should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 103:

```
address token0,
address token1,
```

### 3.104 CVF-104

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

**Recommendation** The argument type should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 104:

```
function withdrawFees(address rewardToken) external
    ↪ onlyRevenueController {
```

### 3.105 CVF-105

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

**Description** Here the accumulated fees are always sent to the revenue controller and are always sent in a whole.
**Recommendation** Consider allowing a caller to specify the destination address and the amount to be withdrawn.
**Client Comment** We would only want to withdraw all fees.

Listing 105:

```
IERC20(rewardToken).safeTransfer(msg.sender, fees);
```

## 3.106 CVF-106

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

**Description** The state is updated after an untrusted external call, which could make a reentrancy attack possible allowing the revenue controller to withdraw more tokens than it should be allowed to withdraw. Note, that some tokens, such as those implementing ERC-777 could call the recipient contract when tokens are transferred.
**Recommendation** Consider updating state before calling untrusted code.

Listing 106:

```
377 IERC20(rewardToken).safeTransfer(msg.sender, fees);
    rewardFeesTotal[rewardToken] = 0;
```

## 3.107 CVF-107

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

**Description** These calls should be executed only when the corresponding amounts are not zero.
**Client Comment** This was refactored.

Listing 107:

```
400 token0.safeTransfer(msg.sender, token0FeeAmount);
    token1.safeTransfer(msg.sender, token1FeeAmount);
```

## 3.108 CVF-108

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

**Recommendation** It is a good practice to put a comment into an empty block to explain why the block is empty.
**Client Comment** Decided to leave it as it is.

Listing 108:

```
412 receive() external payable {}
```

## 3.109 CVF-109

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

**Recommendation** The argument types should be "ICLR" and "IStakedCLRToken" respectively.
**Client Comment** Decided to leave it as address.

Listing 109:

```
constructor(address _clrImplementation, address
    ↪ _sclrTokenImplementation) {
```

## 3.110 CVF-110

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

**Recommendation** The constructor should emit the "CLRImplementationSet" and "CLRTokenImplementationSet" events to make it easier to track the current implementation by event flow.

Listing 110:

```
constructor(address _clrImplementation, address
    ↪ _sclrTokenImplementation) {
```

## 3.111 CVF-111

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

**Recommendation** The arguments type should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 111:

```
function deployCLRPool(address _proxyAdmin)

function deploySCLRToken(address _proxyAdmin)
```

## 3.112 CVF-112

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

**Recommendation** The returned type should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 112:

```
returns (address pool)
```

## 3.113 CVF-113

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

**Recommendation** The returned type should be "IStakedCLRToken".
**Client Comment** Decided to leave it as address.

Listing 113:

```
returns (address token)
```

## 3.114 CVF-114

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

**Recommendation** The argument type should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 114:

```
function setCLRImplementation(address _clrImplementation)
```

## 3.115 CVF-115

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

**Description** These events are emitted event if the corresponding implementation didn't actually change.
**Client Comment** Decided to leave it as it is.

Listing 115:

```
51 emit CLRImplementationSet(_clrImplementation);

59 emit CLRTokenImplementationSet(_sCLRTokenImplementation);
```

## 3.116 CVF-116

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

**Recommendation** The argument type should be "IStakedCLRToken".
**Client Comment** Decided to leave it as address

Listing 116:

```
54 function setsCLRTokenImplementation(address
    ↪ _sCLRTokenImplementation)
```

## 3.117 CVF-117

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

**Recommendation** Events are usually named via nouns, such as "CLRImplementation", "CLTTokenImplementation".
**Client Comment** Decided to leave it as it is.

Listing 117:

```
64 event CLRImplementationSet(address indexed clrImplementation);

66 event CLRTokenImplementationSet(address indexed
    ↪ sCLRTokenImplementation);
```

## 3.118 CVF-118

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

**Recommendation** The type of this variable should be "IERC20 []".
**Client Comment** Decided to leave it as address[].

Listing 118:

```
address[] public rewardTokens;
```

## 3.119 CVF-119

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

**Recommendation** The key type for these mappings should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 119:

```
mapping(address => bool) public isRewardContract;

mapping(address => uint256) public clrPoolVestingPeriod;
```

## 3.120 CVF-120

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

**Recommendation** It would be more efficient to merge these mappings into a single mapping whose keys are CLR pools and values are struct with three fields encapsulating the values of the original mappings.
**Client Comment** Decided to leave it as it is.

Listing 120:

```
mapping(address => bool) public isRewardContract;

mapping(address => mapping(address => mapping(address => uint256
  ↪ [2][])))
    public vestingSchedules;

mapping(address => uint256) public clrPoolVestingPeriod;
```

## 3.121 CVF-121

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

**Recommendation** The type of this mapping should be: "mapping(ICLR => mapping (IERC20 => mapping (address => uint256[2][])))".
**Client Comment** Decided to leave it as address

Listing 121:

```
mapping(address => mapping(address => mapping(address => uint256
    ↪ [2][])))
```

## 3.122 CVF-122

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

**Recommendation** A struct of two fields would be more efficient than a fixed-size array of two elements, as it would not require length checks on access.
**Client Comment** Decided to leave it as it is.

Listing 122:

```
mapping(address => mapping(address => mapping(address => uint256
    ↪ [2][])))
```

## 3.123 CVF-123

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

**Recommendation** The first key type for these mappings should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 123:

```
mapping(address => mapping(address => uint256))

mapping(address => mapping(address => uint256))

mapping(address => uint256) public totalEscrowedBalance;
```

## 3.124 CVF-124

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

**Recommendation** It would be more efficient to merge these three mappings into a single mapping whose keys are reward tokens and values are struct of three fields encapsulating the values of the original mappings.
**Client Comment** Decided to leave it as it is.

Listing 124:

```
mapping(address => mapping(address => uint256))
    public totalEscrowedAccountBalance;

mapping(address => mapping(address => uint256))
    public totalVestedAccountBalance;

mapping(address => uint256) public totalEscrowedBalance;
```

## 3.125 CVF-125

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

**Recommendation** The type of the "_rewardsContract" and "pool" arguments should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 125:

```
64  function addRewardsContract(address _rewardContract) external onlyOwner {

72  function removeRewardsContract(address _rewardContract) external onlyOwner {

110 function setCLRPoolVestingPeriod(address pool, uint256 vestingPeriod)

142     address pool,

154     address pool,

166     address pool,

178     address pool,

193     address pool,

210     address pool,

225     address pool,

236     address pool,

247     address pool,

289     address pool,

334 function vest(address pool, address token) external {
```

## 3.126 CVF-126

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

**Description** Function arguments are named "rewardsContract" while corresponding event parameters are named "rewardContract".
**Recommendation** Consider using consistent naming.
**Client Comment** Decided to leave it as it is.

Listing 126:

```
function addRewardsContract(address _rewardContract) external
    ↪ onlyOwner {

event RewardContractAdded(address indexed rewardContract);
```

## 3.127 CVF-127

- **Severity** Major
- **Category** Unclear behavior
- **Status** Info
- **Source** RewardEscrow.sol

**Description** The event is emitted even if the rewards contract was already added.

Listing 127:

```
emit RewardContractAdded(_rewardContract);
```

## 3.128 CVF-128

- **Severity** Major
- **Category** Unclear behavior
- **Status** Info
- **Source** RewardEscrow.sol

**Description** The event is emitted even if the rewards contract wasn't added.

Listing 128:

```
emit RewardContractRemoved(_rewardContract);
```

## 3.129 CVF-129

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

**Recommendation** The type of the "rewardToken" and "token" arguments should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 129:

```
function addRewardsToken(address rewardToken) external onlyOwner {

function removeRewardsToken(address rewardToken) external onlyOwner {

function balanceOf(address token, address account)

function totalSupply(address token) external view returns (uint256) {

    address token,

    address token,

    address token,

    address token,

    address token,

    address token,

    address token,

    address token,

    address token,

    address token,

function vest(address pool, address token) external {
```

### 3.130 CVF-130

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

**Description** These loops doesn't scale.
**Recommendation** Consider maintaining a mapping from reward token addresses to the indexes of these reward tokens in the "rewardTokens" array.
**Client Comment** Removed "addRewardsToken" and "removeRewardsToken", as well as getters.

Listing 130:

```
81 for (uint256 i = 0; i < rewardTokens.length; ++i) {

94 for (uint256 i = 0; i < rewardTokens.length; ++i) {
```

### 3.131 CVF-131

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

**Description** This function allows setting a vesting period for a pool that is not added as a rewards contract. Probably not an issue.
**Client Comment** Not an issue.

Listing 131:

```
114 clrPoolVestingPeriod[pool] = vestingPeriod;
```

### 3.132 CVF-132

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

**Description** This event is emitted even if the vesting period didn't actually change.
**Client Comment** Not an issue.

Listing 132:

```
115 emit VestingPeriodSet(pool, vestingPeriod);
```

## 3.133 CVF-133

- **Severity** Moderate
- **Category** Suboptimal
- **Status** Info
- **Source** RewardEscrow.sol

**Description** The obtained index is guaranteed to point to the next (i.e. earliest) vesting entry only if vesting entries are ordered by time, which could not be the case if entries were added with different vesting periods.
**Recommendation** If this is not an issue, consider explaining this fact in the comment, otherwise consider addressing this issue in some way.
**Client Comment** The Vesting Period is immutable once a pool is deployed.

Listing 133:

```
190 * @notice Obtain the index of the next schedule entry that will
    ↪ vest for a given user.
```

## 3.134 CVF-134

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

**Description** This loop doesn't scale.
**Recommendation** Consider storing the next vesting index per pool+token+account combination.

Listing 134:

```
198 for (uint256 i = 0; i < len; i++) {
```

## 3.135 CVF-135

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

**Recommendation** The size of this array should be derived from the "MAX_VESTING_ENTRIES" constant.
**Client Comment** Fixed by returning a dynamic array.

Listing 135:

```
250 ) external view returns (uint256[520] memory) {
```

## 3.136 CVF-136

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

**Description** Returning a large fixed-size array is suboptimal as in most cases not all the element will be filled.
**Recommendation** Consider returning a dynamic array.
**Client Comment** Fixed by returning a dynamic array.

Listing 136:

```
250 ) external view returns (uint256[520] memory) {
```

## 3.137 CVF-137

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

**Description** The "totalEscrowedBalance[token]" value that were just written into the storage is read from the storage again.
**Recommendation** Consider caching and reusing the written value.
**Client Comment** Decided to leave it as it is.

Listing 137:

```
293 totalEscrowedBalance[token] = totalEscrowedBalance[token].add(
      ↪ quantity);

295     totalEscrowedBalance[token] <=
```

## 3.138 CVF-138

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

**Description** In case this account already have some vestings for this token in another pools, this will overwrite the total escrowed balance, stored for this token+account combination.
**Recommendation** Consider always adding "quantity" to the existing value.

Listing 138:

```
311 totalEscrowedAccountBalance[token][account] = quantity;
```

## 3.139 CVF-139

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

**Description** It is nor guaranteed that "time" here is not before the tie of the latest existing entry, as the vesting period may change.
**Recommendation** Consider adding an explicit require statement to guarantee that entries are recorded in order.
**Client Comment** The Vesting Period is immutable once a pool is deployed.

Listing 139:

```
vestingSchedules[pool][token][account].push([time, quantity]);
```

## 3.140 CVF-140

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

**Description** This loop iterates on already vested entries.
**Recommendation** Consider storing the index of the earliest not vested entry and starting iterating from this index.
**Client Comment** Decided to leave it as it is.

Listing 140:

```
for (uint256 i = 0; i < numEntries; i++) {
```

## 3.141 CVF-141

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

**Recommendation** Events are usually named via nouns, such as "Vesting", "NewVestingEntry" etc.
**Client Comment** Decided to leave it as it is.

Listing 141:

```
event Vested(

event VestingEntryCreated(

event RewardContractAdded(address indexed rewardContract);

event RewardContractRemoved(address indexed rewardContract);

event RewardTokenAdded(address indexed rewardToken);

event RewardTokenRemoved(address indexed rewardToken);

event VestingPeriodSet(address indexed pool, uint256 vestingPeriod);
```

## 3.142 CVF-142

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

**Recommendation** The type of the "pool" and "rewardContract" parameters should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 142:

```
    address indexed pool,

    address indexed pool,

event RewardContractAdded(address indexed rewardContract);

event RewardContractRemoved(address indexed rewardContract);

event VestingPeriodSet(address indexed pool, uint256 vestingPeriod);
```

### 3.143 CVF-143

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

**Recommendation** The type of the "token" and "rewardToken" parameters should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 143:

```
    address indexed token,

    address indexed token,

    address indexed token,

event RewardTokenAdded(address indexed rewardToken);

event RewardTokenRemoved(address indexed rewardToken);
```

### 3.144 CVF-144

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

**Description** This function and modifier are almost always used together.
**Recommendation** Consider merging them into a single modifier checking that address is not locked and then locks it.
**Client Comment** Decided to leave it as it is.

Listing 144:

```
function lock(address _address) internal {

modifier notLocked(address lockedAddress) {
```

### 3.145 CVF-145

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

**Recommendation** The key type for these mappings should be "IERC20".
**Client Comment** Decided to leave it as address

Listing 145:

```
mapping(address => uint256) public lastUpdateTime; // last time
    ↪ the rewards have been updated

mapping(address => RewardInformation) public rewardInfo;
```

### 3.146 CVF-146

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

**Description** The number format of this field is unclear.
**Recommendation** Consider documenting.

Listing 146:

```
uint256 rewardRate; // reward amount unlocked per second
```

### 3.147 CVF-147

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

**Recommendation** This function wouldn't be necessary if the corresponding variable would be public and owuld have a proper name.
**Client Comment** Decided to leave it as it is.

Listing 147:

```
function stakedTotalSupply() external view override returns (
    ↪ uint256) {
```

## 3.148 CVF-148

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

**Recommendation** This function wouldn't be necessary if the corresponding mapping would be public and would have a proper name.
**Client Comment** Decided to leave it as it is.

Listing 148:

```
function stakedBalanceOf(address account)
```

## 3.149 CVF-149

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

**Recommendation** The type of the "token" argument should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 149:

```
function rewardPerToken(address token)

function earned(address account, address token)

function getRewardForDuration(address token)

function claimRewardForSingleToken(address token) private {

function initializeReward(uint256 rewardAmount, address token)

function updateReward(address account, address token) private {
```

## 3.150 CVF-150

- **Severity** Minor
- **Category** Overflow/Underflow
- **Status** Info
- **Source** StakingRewards.sol

**Description** Phantom overflow is possible here.
**Recommendation** Consider using the "muldiv" function.

Listing 150:

```
            .mul(rewardInfo[token].rewardRate)
            .mul(1e18)
            .div(_stakedTotalSupply)

_stakedBalances[account]
      .mul(

      )
      .div(1e18)
```

## 3.151 CVF-151

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

**Recommendation** The denominator value should be a named constant.
**Client Comment** Decided to leave it as it is.

Listing 151:

```
.mul(1e18)
```

## 3.152 CVF-152

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

**Description** This function claims rewards in all the tokens, which could be gas consuming.
**Recommendation** Consider allowing a user to claim rewards in certain tokens only, e.g. by providing a bit mask.
**Client Comment** The current UI design and UX doesn't allow for separate token withdrawals.

Listing 152:

```
function claimReward() public override {
```

## 3.153 CVF-153

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

**Description** If transfer for one token will fail, the whole transaction will be reverted and the user will get no tokens.
**Recommendation** Consider allowing a user to claim retards in different tokens separately.
**Client Comment** Cases where reward token transfers fail should be rare, only for tokens which have overriden their transfer functions with custom logic. Using proper ERC-20 tokens as reward tokens will be the responsibility of the pool sponsor.

Listing 153:

```
201 claimRewardForSingleToken(rewardTokens[i]);
```

## 3.154 CVF-154

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

**Description** Inside this call, the "_stakedBalances[account]" expression is calculated on every loop iteration.
**Recommendation** Consider refactoring the code to avoid multiple calculation of the same values.
**Client Comment** Decided to leave it as it is.

Listing 154:

```
201 claimRewardForSingleToken(rewardTokens[i]);
```

## 3.155 CVF-155

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

**Description** Currently, in case a user cannot take his reward in a whole, no reward will be given to the user at all.
**Recommendation** Consider giving at least what can be given.

Listing 155:

```
213 rewardInfo[token].remainingRewardAmount >= rewardAmount
```

## 3.156 CVF-156

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

**Description** Rewards for previous periods as well as for the current unfinished period (if any) are not taken into account.
**Recommendation** Should be: rewardTokenInfo.totalRewardAmount = rewardTokenInfo.totalRewardAmount.add (rewardAmount);
**Client Comment** Value tracks only the total reward amounts for the latest initialized reward program. Modified comment to reflect that.

Listing 156:

```
263 rewardTokenInfo.totalRewardAmount = rewardAmount;
```

## 3.157 CVF-157

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

**Description** The remaining rewards for previous periods and for the current unfinished period (if any) are not taken into account here, so user will not be able to claim all the rewards, as the remaining rewards amount will drop to zero before rewards for all the periods will be claimed. Should be: rewardTokenInfo.remainingRewardAmount = rewardTokenInfo.remainingRewardAmoun.add (rewardAmount);

Listing 157:

```
264 rewardTokenInfo.remainingRewardAmount = rewardAmount;
```

## 3.158 CVF-158

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

**Description** There are no range checks for the argument.
**Recommendation** Consider adding appropriate checks.

Listing 158:

```
275 function setRewardsDuration(uint256 _rewardsDuration)
```

## 3.159 CVF-159

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

**Recommendation** The event is emitted even if the rewards duration didn't change.
**Client Comment** Decided to leave it as it is.

Listing 159:

```
emit RewardsDurationUpdated(rewardsDuration);
```

## 3.160 CVF-160

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

**Recommendation** This function should emit some event.
**Client Comment** Decided to leave it as it is.

Listing 160:

```
function setRewardsAreEscrowed(bool _rewardsAreEscrowed)
```

## 3.161 CVF-161

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

**Recommendation** Events are usually named via nouns,m such as "NewReward", "Stake", "Withdrawal" etc.
**Client Comment** Decided to leave it as it is.

Listing 161:

```
event RewardAdded(uint256 reward);
event Staked(address indexed user, uint256 amount);
event Withdrawn(address indexed user, uint256 amount);
event RewardClaimed(

event RewardsDurationUpdated(uint256 newDuration);
event Recovered(address token, uint256 amount);
```

### 3.162 CVF-162

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

**Recommendation** The type of the "token" parameter should be "IERC20".
**Client Comment** Decided to leave it as address

Listing 162:

```
331       address indexed token ,

335 event Recovered(address token , uint256 amount);
```

### 3.163 CVF-163

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

**Description** This event is never emitted.
**Recommendation** Consider removing it.
**Client Comment** Removed event in minor fixes PR.

Listing 163:

```
335 event Recovered(address token , uint256 amount);
```

### 3.164 CVF-164

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

**Recommendation** The type of the "_proxyAdmin" argument should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 164:

```
7 constructor(address _logic , address _proxyAdmin)
```

## 3.165 CVF-165

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

**Description** There is no access level specified for this mapping, so internal access will be used by default.
**Recommendation** Consider explicitly specifying an access level.
**Client Comment** Decided to leave it as it is.

Listing 165:

```
15  mapping(address => address) proxyAdmins;
```

## 3.166 CVF-166

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

**Recommendation** This value could be obtained as: abi.encodeWithSignature ("implementation()")
**Client Comment** Decided to leave it as it is.

Listing 166:

```
33  hex"5c60da1b"
```

## 3.167 CVF-167

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

**Recommendation** This value could be obtained as: abi.encodeWithSignature ("admin()")
**Client Comment** Decided to leave it as it is.

Listing 167:

```
55  hex"f851a440"
```

## 3.168 CVF-168

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

**Recommendation** This function should log an event.
**Client Comment** Decided to leave it as it is.

Listing 168:

```
function transferProxyOwnership(address proxy, address newAdmin)

function addProxyAdmin(address proxy, address admin) external
    ↪ onlyOwner {
```

## 3.169 CVF-169

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

**Description** Despite the name and the comment, this function replaces the current admin with a new one, rather than adds a new admin.
**Recommendation** Consider renaming the function and changing the comment.
**Client Comment** Decided to leave it as it is.

Listing 169:

```
 * Add proxy admin to a given proxy

function addProxyAdmin(address proxy, address admin) external
    ↪ onlyOwner {
```

## 3.170 CVF-170

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

**Description** The name looks like the name of a getter function, while this is actually a modifier.
**Recommendation** Consider renaming to "onlyProxyAdmin".
**Client Comment** Decided to leave it as it is.

Listing 170:

```
modifier isProxyAdmin(address proxy, address user) {
```

### 3.171 CVF-171

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

**Description** The second argument always equals to “msg.sender”.
**Recommendation** Consider removing the argument and using “msg.sender” instead.
**Client Comment** Decided to leave it as it is.

Listing 171:

```
modifier isProxyAdmin(address proxy, address user) {
```

### 3.172 CVF-172

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

**Description** There is no access level specified for this variable, so internal access will be used by default.
**Recommendation** Consider explicitly specifying an access level.
**Client Comment** Decided to leave it as it is.

Listing 172:

```
ICLRDeployer clrDeployer;
```

### 3.173 CVF-173

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

**Recommendation** The type of this argument should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 173:

```
address _proxyAdmin,
```

### 3.174 CVF-174

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

**Recommendation** The type of this argument should be "ICLRDeployer".
**Client Comment** Decided to leave it as address.

Listing 174:

```
address _clrDeployer
```

### 3.175 CVF-175

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

**Description** There is no access modifier specified for this variable, so internal access will be used by default.
**Recommendation** Consider explicitly specifying an access level.
**Client Comment** Decided to leave it as it is.

Listing 175:

```
ICLRDeployer clrDeployer;
```

### 3.176 CVF-176

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

**Recommendation** The type of this argument should be “IProxyAdmin”.
**Client Comment** Decided to leave it as address.

Listing 176:

```
address _proxyAdmin,
```

## 3.177 CVF-177

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

**Recommendation** The type of this argument should be "ICLRDeployer".
**Client Comment** Decided to leave it as address.

Listing 177:

```
address _clrDeployer
```

## 3.178 CVF-178

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

**Recommendation** The type of the "_proxyAdmin" argument should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 178:

```
constructor(address _logic, address _proxyAdmin)
```

## 3.179 CVF-179

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

**Description** There is no length check for the "prices" argument.
**Recommendation** Consider requiring that "prices" contain exactly two elements.
**Client Comment** Function was removed.

Listing 179:

```
function getTWAP(int56[] memory prices, uint32 secondsAgo)
```

## 3.180 CVF-180

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

**Description** Despite the name, the "prices" array doesn't contain prices, but rather cumulative tick over time.
**Recommendation** Consider renaming the argument and/or explaining in the documentation comment the semantics of its values.
**Client Comment** Function was removed.

Listing 180:

```
23 function getTWAP(int56[] memory prices, uint32 secondsAgo)
```

## 3.181 CVF-181

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

**Description** According to general precedence rules, this formula is evaluated as: (1.0001^(currentPrice - pastPrice)) / secondsAgo, i.e. division is performed after the exponentiation.
**Recommendation** Consider rewriting the formula as: 1.0001^((currentPrice - pastPrice) / secondsAgo)
**Client Comment** Function was removed, not using TWAP in the code.

Listing 181:

```
29 // 1.0001 ^ (currentPrice - pastPrice) / secondsAgo
```

## 3.182 CVF-182

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

**Description** This formula doesn't actually calculate a TWAP price in canonical meaning.
**Recommendation** Consider explaining in a comment what kind of TWAP is calculated here.
**Client Comment** Function was removed.

Listing 182:

```
29 // 1.0001 ^ (currentPrice - pastPrice) / secondsAgo
```

## 3.183 CVF-183

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

**Description** A tick is basically the logarithm of a price, not the price itself, so this formula calculates the time-weighted average of price logarithm and them converts this average logarithm into price. Thus, the function actually calculates a time-weighted geometric mean rather than time-weighted average.
**Recommendation** Consider explaining thins in a documentation comment.
**Client Comment** Function was removed. Uni V3 Uses time-weighted geometric mean.

Listing 183:

```
// 1.0001 ^ (currentPrice − pastPrice) / secondsAgo
```

## 3.184 CVF-184

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

**Recommendation** This constant value could be precalculated.

Listing 184:

```
ABDKMath64x64.divu(10001, 10000),
```

## 3.185 CVF-185

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

**Recommendation** The conversion to “uint256” is redundant as the “toUInt” function already returns “uint256”.
**Client Comment** Function removed.

Listing 185:

```
uint256(ABDKMath64x64.toUInt(fraction))
```

## 3.186 CVF-186

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

**Description** Precision is lost when converting "fraction" to "uint256".
**Recommendation** Consider performing fractional exponentiation via log+exp. Note, that the "log" part could be precomputed.
**Client Comment** Function removed.

Listing 186:

```
uint256(ABDKMath64x64.toUInt(fraction))
```

## 3.187 CVF-187

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

**Description** This function returns the absolute swap amount but doesn't return its sign, so the caller has to do additional work to figure out the sign, while the sign was already implicitly calculated inside this function.
**Recommendation** Consider returning the swap amount as a signed integer, or returning the sign separately from the absolute swap amount.

Listing 187:

```
function calculateSwapAmount(
```

## 3.188 CVF-188

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

**Recommendation** // X * T
**Client Comment** Implemented as suggested.

Listing 188:

```
uint256 mul1 = amountsMinted.amount0ToMint.mul(
```

## 3.189 CVF-189

- **Severity** Major
- **Category** Overflow/Underflow
- **Status** Fixed
- **Source** Utils.sol

**Description** These multiplications may overflow.
**Recommendation** Consider using 512-bits multiplications or limit the input amounts at 128 bits.
**Client Comment** Using SafeMath library, so the function will revert if the values overflow. In that case the caller of the function will have to adjust the amounts manually.

Listing 189:

```
69 uint256 mul1 = amountsMinted.amount0ToMint.mul(

72 uint256 mul2 = amountsMinted.amount1ToMint.mul(
```

## 3.190 CVF-190

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

**Recommendation** // Y * Z
**Client Comment** Implemented as suggested.

Listing 190:

```
72 uint256 mul2 = amountsMinted.amount1ToMint.mul(
```

## 3.191 CVF-191

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

**Recommendation** // X * T - Y * Z
**Client Comment** Implemented as suggested.

Listing 191:

```
75 uint256 sub = subAbs(mul1, mul2);
```

## 3.192 CVF-192

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

**Recommendation** // I * Y
**Client Comment** Implemented as suggested.

Listing 192:

```
uint256 add1 = ABDKMath64x64.mulu(
```

## 3.193 CVF-193

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

**Recommendation** // p0 * I * X
**Client Comment** Implemented as suggested.

Listing 193:

```
uint256 add2 = midPrice
```

## 3.194 CVF-194

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

**Recommendation** // p0 * Z
**Client Comment** Implemented as suggested.

Listing 194:

```
uint256 add3 = midPrice.mul(amountsMinted.amount0Minted).div(1
    ↪ e12);
```

## 3.195 CVF-195

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

**Recommendation** // l * Y + p0 *l * X + p0 * Z + T
**Client Comment** Implemented as suggested.

Listing 195:

```
86 uint256 add = add1.add(add2).add(add3).add(amountsMinted.
     ↪ amount1Minted);
```

## 3.196 CVF-196

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

**Description** The schema with square roots looks cumbersome and inefficient.
**Recommendation** We would suggest doing calculations in the following way: int128 midPrice64x64 = ABDKMath64x64.divu (midPrice, 1e12); uint256 denominator = ABDKMath64x64.mulu (ABDKMath64x64.mul (midPrice64x64, liquidityRatio), amountsMinted.amount0ToMint). add (ABDKMath64x64.mulu (midPrice64x64, amountsMinted.amount0Minted)). add (ABDKMath64x64.mulu (liquidityRatio, amountsMinted.amount1ToMint)). add (amountsMinted.amount1Minted); uint256 a = muldiv (amountsMinted.amount0ToMint, amountsMinted.amount1Minted, denominator); uint256 b = muldiv (amountsMinted.amount1ToMint, amountsMinted.amount0Minted, denominator); return a >= b ? a - b : b - a; Here the “muldiv” function is taken from here: https://xn–2-umb.com/21/muldiv/index.html
**Client Comment** Implemented as suggested.

Listing 196:

```
88 // Some numbers are too big to fit in ABDK's div 128-bit
     ↪ representation
   // So calculate the root of the equation and then raise to the 2
     ↪ nd power
```

## 3.197 CVF-197

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

**Recommendation** This function could be simplified as: return (a - time - 1) <= (b - time - 1);
**Client Comment** Function removed.

Listing 197:

```
100 function lte(
```

## 3.198 CVF-198

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

**Description** Safe subtractions are redundant here as the condition prevents underflow.
**Recommendation** Consider using plain subtractions.
**Client Comment** Implemented as suggested.

Listing 198:

```
119 return amount0 >= amount1 ? amount0.sub(amount1) : amount1.sub(
      ↪ amount0);
```

## 3.199 CVF-199

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

**Recommendation** These functions should emit some events and these events should be defined in this interface.
**Client Comment** They emit events, the events are not necessary in the interface though.

Listing 199:

```
8 function addManager(address manager, address fund) external;

13 function removeManager(address manager, address fund) external;

26 function setRevenueController(address controller) external;
```

### 3.200 CVF-200

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

**Recommendation** The type of this argument should be “ICLR”.
**Client Comment** Decided to leave it as address.

Listing 200:

```
address _clrPool,
```

### 3.201 CVF-201

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

**Description** It is uncommon to name functions IN_UPPER_CASE even when they return constant values.
**Recommendation** Consider naming 'inCamelCase'.
**Client Comment** Decided to leave it as it is.

Listing 201:

```
function MAX_VESTING_ENTRIES() external view returns (uint256);
```

## 3.202 CVF-202

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

**Recommendation** The type of the “token” and “rewardToken” arguments should be “IERC20”.
**Client Comment** Decided to leave them as address.

Listing 202:

```
function addRewardsToken(address rewardToken) external;

    address token,

function balanceOf(address token, address account)

function getNextVestingIndex(address token, address account)

function getNextVestingQuantity(address token, address account)

function getNextVestingTime(address token, address account)

    address token,

    address token,

function numVestingEntries(address token, address account)

function removeRewardsToken(address rewardToken) external;

function setCLRPoolVestingPeriod(address rewardToken, uint256
   ↪ vestingPeriod)

function totalSupply(address token) external view returns (
   ↪ uint256);
```

## 3.203 CVF-203

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

**Recommendation** The type of the “pool” argument should be “ICLR”.
**Client Comment** Decided to leave it as address.

Listing 203:

```
address pool,
```

## 3.204 CVF-204

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

**Recommendation** These functions should be moved to a separate "Ownable" interface.
**Client Comment** Decided to leave it as it is.

Listing 204:

```
function owner() external view returns (address);

function renounceOwnership() external;

function transferOwnership(address newOwner) external;
```

## 3.205 CVF-205

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

**Recommendation** The return type should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 205:

```
function rewardTokens(uint256) external view returns (address);
```

## 3.206 CVF-206

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

**Description** Despite the name, this function returns a single token, rather than several of them.
**Recommendation** Consider renaming to "rewardToken".

Listing 206:

```
function rewardTokens(uint256) external view returns (address);
```

## 3.207 CVF-207

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

**Description** This function is not implemented in the "RewardEscrow" smart contract.
**Recommendation** Consider either removing this function here or implementing there.

Listing 207:

```
function rewardsTokenVestingPeriod(address) external view returns (uint256);
```

## 3.208 CVF-208

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

**Description** The semantics of the arguments and the returned value is unclear.
**Recommendation** Consider giving them descriptive names and/or adding a documentation comment.
**Client Comment** Decided to leave it as it is.

Listing 208:

```
function rewardsTokenVestingPeriod(address) external view returns (uint256);

function totalEscrowedAccountBalance(address, address)

    returns (uint256);

function totalEscrowedBalance(address) external view returns (uint256);

function totalVestedAccountBalance(address, address)

    returns (uint256);

function vestingSchedules(
    address,
    address,
    uint256,
    uint256
) external view returns (uint256);
```

## 3.209 CVF-209

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

**Recommendation** The type of the “token” argument should be “IERC20”.
**Client Comment** Decided to leave it as address.

Listing 209:

```
function rewardPerToken(address token) external view returns (uint256);

function earned(address account, address token)

function getRewardForDuration(address token)

function initializeReward(uint256 rewardAmount, address token) external;
```

## 3.210 CVF-210

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

**Recommendation** The return type should be “IERC20 []”.
**Client Comment** Decided to leave it as address[].

Listing 210:

```
function getRewardTokens() external view returns (address[] memory);
```

## 3.211 CVF-211

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

**Recommendation** The return type should be “IERC20”.
**Client Comment** Decided to leave it as address.

Listing 211:

```
function rewardTokens(uint256 index) external view returns (address);
```

## 3.212 CVF-212

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

**Description** Despite the name, this function returns a single token.
**Recommendation** Consider renaming.
**Client Comment** Decided to leave it as it is.

Listing 212:

```
function rewardTokens(uint256 index) external view returns (
   ↪ address);
```

## 3.213 CVF-213

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

**Recommendation** This function should return the amounts of reward tokens claimed.
**Client Comment** Decided to leave it as it is.

Listing 213:

```
function claimReward() external;
```

## 3.214 CVF-214

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

**Recommendation** These functions should emit some events and these events should be defined in this interface.
**Client Comment** Decided to leave it as it is.

Listing 214:

```
function addProxyAdmin(address proxy, address admin) external;

function changeProxyAdmin(address proxy, address newAdmin)
   ↪ external;

function upgrade(address proxy, address implementation) external
   ↪ ;
```

## 3.215 CVF-215

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

**Recommendation** These functions should be moved to a separate "Ownable" interface.
**Client Comment** Decided to leave it as it is.

Listing 215:

```
function owner() external view returns (address);

function renounceOwnership() external;

function transferOwnership(address newOwner) external;
```

## 3.216 CVF-216

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

**Description** This function is not implemented by the "LNTerminal" smart contract.
**Recommendation** Consider either removing the function from here or implementing there.
**Client Comment** Fixed

Listing 216:

```
function claimReward(address clrPool) external;
```

## 3.217 CVF-217

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

**Recommendation** The type of the "clrPool" arguments should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 217:

```
9  function claimReward(address clrPool) external;

39     address clrPool,

46     address clrPool,

53     address clrPool,

58 function removeLiquidity(address clrPool, uint256 amount)
     ↪ external;

60 function removeLiquidityAndClaimReward(address clrPool, uint256
     ↪ amount)
```

## 3.218 CVF-218

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

**Recommendation** The return type should be "ICLRDeployer".
**Client Comment** Decided to leave it as address.

Listing 218:

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

## 3.219 CVF-219

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

**Recommendation** The type of the "proxyAdmin" argument should be "IProxyAdmin".
**Client Comment** Decided to leave it as address.

Listing 219:

```
18 address proxyAdmin
```

### 3.220 CVF-220

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

**Recommendation** The type for these arguments should be “IERC20”.
**Client Comment** Decided to leave it as address.

Listing 220:

```
address token0,
address token1,

address token0,
address token1,
```

### 3.221 CVF-221

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

**Description** The number formats for these arguments are unclear.
**Recommendation** Consider documenting.

Listing 221:

```
uint24 fee,
uint160 initPrice
```

### 3.222 CVF-222

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

**Recommendation** The return type should be “IUniswapV3Pool”.
**Client Comment** Decided to leave it as address.

Listing 222:

```
) external returns (address pool);
```

## 3.223 CVF-223

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

**Recommendation** The return type should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 223:

```
function deployedCLRPools(uint256) external view returns (
    ↪ address);

) external view returns (address pool);
```

## 3.224 CVF-224

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

**Description** Despite the name, this function returns a single CLR pool, rather than several of them.
**Recommendation** Consider renaming to "deployedCLRPool".
**Client Comment** Decided to leave it as address.

Listing 224:

```
function deployedCLRPools(uint256) external view returns (
    ↪ address);
```

## 3.225 CVF-225

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

**Description** The number format of the returned value is unclear.
**Recommendation** Consider documenting.

Listing 225:

```
function deploymentFee() external view returns (uint256);
```

## 3.226 CVF-226

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

**Description** These two functions have very similar names.
**Recommendation** Consider explaining the difference between then in a documentation.

Listing 226:

```
function initiateNewRewardsProgram(

function initiateRewardsProgram(
```

## 3.227 CVF-227

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

**Recommendation** The return type should be “INonfungiblePositionManager”.
**Client Comment** Decided to leave it as address.

Listing 227:

```
function positionManager() external view returns (address);
```

## 3.228 CVF-228

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

**Recommendation** The type of this argument should be a enum with two valid values or even “bool”.
**Client Comment** Decided to leave it as it is.

Listing 228:

```
uint8 inputAsset,
```

### 3.229 CVF-229

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

**Recommendation** These functions should return the amounts of assets returned.
**Client Comment** Decided to leave it as it is.

Listing 229:

```
58 function removeLiquidity(address clrPool, uint256 amount)
     ↪ external;

60 function removeLiquidityAndClaimReward(address clrPool, uint256
     ↪ amount)
      external;
```

### 3.230 CVF-230

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

**Recommendation** The return type should be "IRewardEscrow".
**Client Comment** Decided to leave it as address.

Listing 230:

```
63 function rewardEscrow() external view returns (address);
```

### 3.231 CVF-231

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

**Description** The number format of the returned values is unclear.
**Recommendation** Consider documenting.
**Client Comment** Decided to leave it as it is. More detailed comments are in Terminal.sol

Listing 231:

```
65 function rewardFee() external view returns (uint256);

69 function tradeFee() external view returns (uint256);
```

## 3.232 CVF-232

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

**Recommendation** The types of these return values should be “ISwapRouter”, “iQuoter”, and “INonfungiblePositionManager” respectively.
**Client Comment** Decided to leave them as address.

Listing 232:

```
address router,
address quoter,
address _positionManager
```

## 3.233 CVF-233

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

**Description** This name is prefixed with underscore (‘_’) while other names of returned values are not prefixed.
**Recommendation** Consider using a consistent naming strategy.
**Client Comment** Decided to leave it as it is.

Listing 233:

```
address _positionManager
```

## 3.234 CVF-234

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

**Recommendation** The return type should be “IUniswapV3Factory”.
**Client Comment** Decided to leave it as address.

Listing 234:

```
function uniswapFactory() external view returns (address);
```

## 3.235 CVF-235

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

**Recommendation** The argument type should be “ICLR”.
**Client Comment** Decided to leave it as address.

Listing 235:

```
function withdrawClaimFees(address pool) external;
```

## 3.236 CVF-236

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

**Recommendation** The argument type should be “IERC20”.
**Client Comment** Decided to leave it as address.

Listing 236:

```
function withdrawFees(address rewardToken) external;
```

## 3.237 CVF-237

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

**Recommendation** The return type should be “IxTokenManager”.
**Client Comment** Decided to leave it as address.

Listing 237:

```
function xTokenManager() external view returns (address);
```

## 3.238 CVF-238

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

**Recommendation** The type of this field should be “IERC20 []”.
**Client Comment** Decided to leave it as address[].

Listing 238:

```
address[] rewardTokens;
```

### 3.239 CVF-239

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

**Recommendation** The type of these fields should be "IERC20".
**Client Comment** Decided to leave it as address.

Listing 239:

```
address token0;
address token1;
```

### 3.240 CVF-240

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

**Recommendation** The return type should be "ICLR".
**Client Comment** Decided to leave it as address.

Listing 240:

```
function clrImplementation() external view returns (address);

function deployCLRPool(address _proxyAdmin) external returns (address pool);
```

### 3.241 CVF-241

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

**Recommendation** The type of the "_proxyAdmin" arguments should be "IProxyAdmin".
**Client Comment** Decided to leave them as address.

Listing 241:

```
function deployCLRPool(address _proxyAdmin) external returns (address pool);

function deploySCLRToken(address _proxyAdmin)
```

### 3.242 CVF-242

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

**Recommendation** The return type should be “IStakedCLRToken”.
**Client Comment** Decided to leave it as address.

Listing 242:

```
    returns (address token);

function sCLRTokenImplementation() external view returns (
  ↪ address);
```

### 3.243 CVF-243

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

**Recommendation** These functions should be moved to a separate “Ownable” interface.

Listing 243:

```
function owner() external view returns (address);

function renounceOwnership() external;

function transferOwnership(address newOwner) external;
```

### 3.244 CVF-244

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

**Recommendation** The argument type should be “ICLR”.
**Client Comment** Decided to leave it as address.

Listing 244:

```
function setCLRImplementation(address _clrImplementation)
  ↪ external;
```

## 3.245 CVF-245

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

**Recommendation** The argument type should be “IStakedCLRToken”.
**Client Comment** Decided to leave it as address.

Listing 245:

```
function setsCLRTokenImplementation(address
    ↪ _sCLRTokenImplementation)
```

## 3.246 CVF-246

- **Severity** Minor
- **Category** Procedural
- **Status** Info
- **Source** IERC20.sol

**Description** The event parameter names are different from those defined in the standard. Note, that unlike function argument names, event parameter names are part of a contracts' public API.
**Recommendation** Consider specifying event parameter names according to the standard.

Listing 246:

```
event Transfer(address indexed from, address indexed to, uint256
    ↪  value);

      address indexed owner,
      address indexed spender,
      uint256 value
```

## 3.247 CVF-247

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

**Description** This comment is confusing.
**Recommendation** Consider elaborating more about what it is and what its functions do.
**Client Comment** Decided to leave it as it is.

Listing 247:

```
* CLR Interface
```

## 3.248 CVF-248

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

**Description** Names of some arguments are prefixed with underscore ('_'), while other argument names are not prefixed.
**Recommendation** Consider using a consistent naming strategy.
**Client Comment** Decided to leave it as it is.

Listing 248:

```
function adminSwap(uint256 amount, bool _0for1) external;

    bool _0for1,
    bytes memory _oneInchData

function calculateMintAmount(uint256 _amount, uint256
  ↪ totalSupply)

function changePool(address _poolAddress, uint24 _poolFee)
  ↪ external;

    string memory _symbol,
    int24 _tickLower,
    int24 _tickUpper,
    uint256 _tradeFee,
    address _token0,
    address _token1,
    address _stakedToken,
    address _terminal,
    address _uniswapPool,
```

## 3.249 CVF-249

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

**Recommendation** The type of the "inputAsset" should be a enum with two valid values, or even "bool".
**Client Comment** Decided to leave it as it is.

Listing 249:

```
function calculateAmountsMintedSingleToken(uint8 inputAsset,
  ↪ uint256 amount)

    uint8 inputAsset,
```

## 3.250 CVF-250

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

**Description** The semantics of the "_amount" argument is unclear from its name.
**Recommendation** Consider using a more descriptive name and/or adding a documentation comment.
**Client Comment** Comments are pretty descriptive in CLR.

Listing 250:

```
function calculateMintAmount(uint256 _amount, uint256
  ↪ totalSupply)
```

## 3.251 CVF-251

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

**Description** This function is not implemented in the "CLR" smart contract.
**Recommendation** Consider either removing it from here or implementing there.

Listing 251:

```
function changePool(address _poolAddress, uint24 _poolFee)
  ↪ external;
```

## 3.252 CVF-252

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

**Recommendation** The type of the "_poolAddress" argument should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 252:

```
function changePool(address _poolAddress, uint24 _poolFee)
  ↪ external;
```

## 3.253 CVF-253

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

**Description** The number format of the "_poolFee" argument is unclear.
**Recommendation** Consider documenting.
**Client Comment** Most of the undocumented logic is related to Uniswap V3 logic, as is this.

Listing 253:

```
function changePool(address _poolAddress, uint24 _poolFee) external;
```

## 3.254 CVF-254

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

**Description** These functions are not a standard and their exact semantics is unclear.
**Recommendation** Consider documenting.
**Client Comment** Functions are from ERC20Upgradeable.

Listing 254:

```
function decreaseAllowance(address spender, uint256 subtractedValue)
    external
    returns (bool);

function increaseAllowance(address spender, uint256 addedValue)
    external
    returns (bool);
```

## 3.255 CVF-255

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

**Description** The semantics of the "amount" arguments is unclear.
**Recommendation** Consider giving more descriptive names to the arguments and/or adding documentation comments.
**Client Comment** Functions were removed.

Listing 255:

```
53 function getAmountInAsset0Terms(uint256 amount)

58 function getAmountInAsset1Terms(uint256 amount)
```

## 3.256 CVF-256

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

**Description** The number formats of the returned values is unclear.
**Recommendation** Consider documenting.
**Client Comment** Functions were removed.

Listing 256:

```
68 function getAsset0Price() external view returns (int128);

70 function getAsset1Price() external view returns (int128);
```

## 3.257 CVF-257

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

**Description** Despite the name, this function returns two balances rather than one.
**Recommendation** Consider renaming to "getBufferTokenBalances".
**Client Comment** Decided to leave it as it is.

Listing 257:

```
78 function getBufferTokenBalance()
```

## 3.258 CVF-258

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

**Description** It is unclear what units the returned value is denominated in.
**Recommendation** Consider documenting.
**Client Comment** Function was removed.

Listing 258:

```
function getNav() external view returns (uint256);
```

## 3.259 CVF-259

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

**Description** The semantics of the returned values is unclear.
**Recommendation** Consider documenting.
**Client Comment** Most of the undocumented logic is related to Uniswap V3 logic, as is this.

Listing 259:

```
function getTicks() external view returns (int24 tick0, int24
    ↪ tick1);
```

## 3.260 CVF-260

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

**Recommendation** The type of these arguments should be “IERC20”.
**Client Comment** Decided to leave it as address.

Listing 260:

```
112     address _token0,
        address _token1,

196 function withdrawToken(address token, address receiver) external
    ↪ ;
```

## 3.261 CVF-261

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

**Recommendation** The type of this argument should be "IStakedCLRToken".
**Client Comment** Decided to leave it as address.

Listing 261:

```
address _stakedToken,
```

## 3.262 CVF-262

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

**Recommendation** The type of the "_uniswapPool" argument should be "IUniswapV3Pool".
**Client Comment** Decided to leave it as address.

Listing 262:

```
address _uniswapPool,
```

## 3.263 CVF-263

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

**Recommendation** These functions should be moved to a separate "Ownable" interface.
**Client Comment** Decided to leave it as it is.

Listing 263:

```
function owner() external view returns (address);

function renounceOwnership() external;

function transferOwnership(address newOwner) external;
```

## 3.264 CVF-264

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

**Description** The semantics of the returned value is unclear.
**Recommendation** Consider giving a descriptive name to the returned value and/or adding a documentation comment.
**Client Comment** Seems pretty straightforward, true if function call succeeded, false if not.

Listing 264:

```
140 function pauseContract() external returns (bool);

194 function unpauseContract() external returns (bool);
```

## 3.265 CVF-265

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

**Description** The number format of the returned value is unclear.
**Recommendation** Consider documenting.
**Client Comment** Not documented because it's the same value as in Uni V3.

Listing 265:

```
144 function poolFee() external view returns (uint24);
```

## 3.266 CVF-266

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

**Description** The semantics and the number format of the argument is unclear.
**Recommendation** Consider documenting.
**Client Comment** This value was removed.

Listing 266:

```
150 function setMaxTwapDeviationDivisor(uint256 newDeviationDivisor)
     ↪ external;
```

## 3.267 CVF-267

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

**Description** These functions seem redundant, as their values could be queried from the tokens directly.
**Client Comment** Pre-calculated values are better than doing math calculations every time.

Listing 267:

```
158 function token0DecimalMultiplier() external view returns (
        ↪ uint256);

160 function token0Decimals() external view returns (uint8);

164 function token1DecimalMultiplier() external view returns (
        ↪ uint256);

166 function token1Decimals() external view returns (uint8);
```

## 3.268 CVF-268

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

**Description** The “decimals” property of a ERC-20 token is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.
**Recommendation** Consider treating all token amounts as integers.
**Client Comment** This was used for xAssetCLR (contract on which CLR is based on) to get TWAP values properly. Will remove this variable everywhere in Terminal.

Listing 268:

```
170 function tokenDiffDecimalMultiplier() external view returns (
        ↪ uint256);
```

## 3.269 CVF-269

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

**Description** The semantics of the returned value is unclear.
**Recommendation** Consider documenting.
**Client Comment** Anyone familiar with Uni V3 should be able to understand what token id stands for related to a pool.

Listing 269:

```
function tokenId() external view returns (uint256);
```

## 3.270 CVF-270

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

**Description** This "is a divisor" is unclear. Does it mean that 200 = 0.5% or 2%?
**Recommendation** Consider giving at least one more example.
**Client Comment** There are other examples throughout the code. You can see UniswapLibrary, there is MINT_BURN_SLIPPAGE = 50, which is equal to 2%.

Listing 270:

```
function tradeFee() external view returns (uint256); // xToken
    ↪ Trade Fee as a divisor (100 = 1%)
```

## 3.271 CVF-271

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

**Recommendation** The type of this field should be "ISwapRouter".
**Client Comment** Decided to leave it as address.

Listing 271:

```
address router;
```

## 3.272 CVF-272

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

**Recommendation** The type of this field should be "IQuoter".
**Client Comment** Decided to leave it as address.

Listing 272:

```
address quoter;
```

## 3.273 CVF-273

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

**Recommendation** The type of this field should be "INonFungiblePositionManager".
**Client Comment** Decided to leave it as address.

Listing 273:

```
address positionManager;
```

## 3.274 CVF-274

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

**Recommendation** The type of this field should be "IERC20 []".
**Client Comment** Decided to leave it as address[].

Listing 274:

```
address[] rewardTokens;
```

## 3.275 CVF-275

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

**Recommendation** The type of this field should be "IRewardEscrow".
**Client Comment** Decided to leave it as address.

Listing 275:

```
address rewardEscrow;
```