Security Assessment

Lixir Finance

Sept 12th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- LBL-01 : Missing Error Messages
- LFL-01 : Missing Error Messages
- LRL-01 : Missing Event Emissions for Significant Transactions
- LRL-02 : Missing Error Messages
- LRL-03 : Lack of Input Validation
- LSS-01 : Redundant Code
- LSS-02 : Lack of Safety Check
- LSS-03 : Missing Error Messages
- LSS-04 : Missing Event Emissions for Significant Transactions
- LSS-05 : Centralization Risks
- LSS-06 : Reentrancy Attack Risks
- LVE-01 : Missing Error Messages
- LVE-02 : Reentrancy Attack Risks
- LVL-01 : Centralization Risks
- LVL-02 : Unhandled Return Values
- LVL-03 : Redundant Contract Import
- LVL-04 : Missing Event Emissions for Significant Transactions
- LVL-05 : Missing Error Messages
- LVL-06 : Incompatibility with Deflationary Tokens
- LVL-07 : Reentrancy Attack Risks
- LVL-08 : Inaccurate Calculation
- LVL-09 : Possibly Uninitialized Variables

Appendix

Disclaimer

About

Summary

This report has been prepared for Lixir to discover issues and vulnerabilities in the source code of the Lixir Finance project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Lixir Finance
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/Lixir-Team/lixir-contracts/tree/master/contracts
Commit	1423c9d8112f103d782bec2bff0558c6be1ca4cc

Audit Summary

Delivery Date	Sept 12, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	1	0	0	0	0	1
Minor	6	0	0	4	1	1
Informational	15	2	0	9	1	3
Discussion	0	0	0	0	0	0

Audit Scope

ID			File	SHA256 Checksum

Understandings

Overview

Lixir Protocol delivers optimal capital efficiency, minimum impermanent loss, and solves the inactive liquidity problem in Uniswap v3 yield farming base on single-side passive rebalancing the liquidity position.

Dependencies

There are a few depending injection contracts or addresses in the current project:

- `token0`, `token1`, `weth9` and `uniV3Factory` for the Lixir Protocol;

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

Priviledged Functions

In the contract `LixirVault`, the certain privileged addresses can operate on the following functions:

- `LixirVault.setMaxSupply()` to update the maximum supply of shares available to users;
- `LixirVault.setPerformanceFee()` to update the applicable fee ratio;
- `LixirVault.setKeeper()` to update the keeper address of the vault contract;
- `LixirVault.setStrategist()` to update the strategist address of the vault contract;
- `LixirVault.emergencyExit()` to withdraw all the tokens from Uniswap and pause the vault contract;
- `LixirVault.unpause()` to unpause the vault.

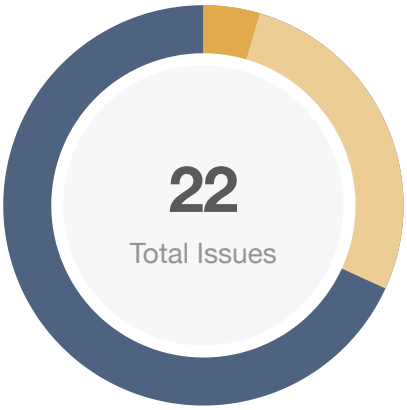
In the contract `LixirStrategySimpleGWAP`, the `strategists` can operate on the following functions:

- `LixirStrategySimpleGWAP.setTickShortDuration()` to update the short duration for checking the recent time weighted tick average;
- `LixirStrategySimpleGWAP.setMaxTickDif()` to update the maximum tolerable difference between ticks;

- `LixirStrategySimpleGWAP.setSpreads()` to update the spreads of the main order and the range order;
- `LixirStrategySimpleGWAP.configureVault()` to update the vault configurations;
- `LixirStrategySimpleGWAP.rebalance()` to rebalance the concentration orders.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	1 (4.55%)
Minor	6 (27.27%)
Informational	15 (68.18%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
LBL-01	Missing Error Messages	Coding Style	Informational	Acknowledged
LFL-01	Missing Error Messages	Coding Style	Informational	Acknowledged
LRL-01	Missing Event Emissions for Significant Transactions	Coding Style	Informational	Resolved
LRL-02	Missing Error Messages	Coding Style	Informational	Acknowledged
LRL-03	Lack of Input Validation	Logical Issue	Informational	Acknowledged
LSS-01	Redundant Code	Coding Style	Informational	Resolved
LSS-02	Lack of Safety Check	Logical Issue	Minor	Acknowledged
LSS-03	Missing Error Messages	Coding Style	Informational	Acknowledged
LSS-04	Missing Event Emissions for Significant Transactions	Coding Style	Informational	Partially Resolved
LSS-05	Centralization Risks	**Centralization / Privilege**	**Minor**	Acknowledged
LSS-06	Reentrancy Attack Risks	Logical Issue	Minor	Acknowledged
LVE-01	Missing Error Messages	Coding Style	Informational	Acknowledged
LVE-02	Reentrancy Attack Risks	Logical Issue	Minor	Acknowledged
LVL-01	Centralization Risks	**Centralization / Privilege**	**Minor**	Partially Resolved

ID	Title	Category	Severity	Status
LVL-02	Unhandled Return Values	Coding Style	Informational	Pending
LVL-03	Redundant Contract Import	Coding Style	Informational	Resolved
LVL-04	Missing Event Emissions for Significant Transactions	Coding Style	Informational	Acknowledged
LVL-05	Missing Error Messages	Coding Style	Informational	Acknowledged
LVL-06	Incompatibility with Deflationary Tokens	Logical Issue	Informational	Acknowledged
LVL-07	Reentrancy Attack Risks	Logical Issue	Informational	Pending
LVL-08	Inaccurate Calculation	Logical Issue	Medium	Resolved
LVL-09	Possibly Uninitialized Variables	Logical Issue	Minor	Resolved

LBL-01 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirBase.sol: 14, 18, 22	Acknowledged

Description

The `require` statements on the aforementioned lines do not have proper error messages implemented.

Recommendation

We recommend adding proper error messages to the `require` statements on the aforementioned lines.

Alleviation

[The Lixir Team]: These are only called by governance or keeper transactions, not users. The vaults have code size issues, so omitting them here is acceptable to us. We may trivially debug via stack trace.

LFL-01 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirFactory.sol: 56~57, 93~94	Acknowledged

Description

The `require` statements on the aforementioned lines do not have proper error messages implemented.

Recommendation

We recommend adding proper error messages to the `require` statements on the aforementioned lines.

Alleviation

[The Lixir Team]: For consistency, we use this from base. Since vault depends also on base, we have code size issues. These are only called by privileged roles, we can debug via stack trace internally.

LRL-01 | Missing Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	Informational	LixirRegistry.sol: 58	Resolved

Description

The function `LixirRegistry.setFeeTo()` affects the sensitive address `feeTo` which indicates where the fee goes so it should emit events as notifications to the users.

Recommendation

We recommend emitting events for the function `LixirRegistry.setFeeTo()`.

Alleviation

The client heeded our advice and resolved the issue by adding a event emission in function `LixirRegistry.setFeeTo()`. The change is reflected in the commit `92b2cfbdb42645f2ede93acbe754ed1020f0ad13`.

LRL-02 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirRegistry.sol: 59	Acknowledged

Description

The `require` statements on the aforementioned line do not have a proper error message implemented.

Recommendation

We recommend adding proper error messages to the `require` statements on the aforementioned lines.

Alleviation

[The Lixir Team]: For consistency, we use this from base. Since vault depends also on base, we have code size issues. These are only called by privileged roles, we can debug via stack trace internally.

LRL-03 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	LixirRegistry.sol: 35~36	Acknowledged

Description

The input addresses `_uniV3Factory` and `_weth9` should never be zero address, so the function `LixirRegistry.constructor()` should have proper input validation.

Recommendation

We recommend adding input validation for the function `LixirRegistry.constructor()`. For example,

```
require(uniV3Factory!= address(0), "uniV3Factory should not be zero address")
require(_weth9 != address(0), "_weth9 should not be zero address")
```

Alleviation

[The Lixir Team]: This is only deployed once, it could also have a nonzero address and still be wrong if there are encoding problems, fine how it is.

LSS-01 | Redundant Code

Category	Severity	Location	Status
Coding Style	Informational	LixirStrategySimpleGWAP.sol: 195	Resolved

Description

The code snippet `diff <= MAX_TICK_DIFF` is repeated on line 195:

```
195         diff <= MAX_TICK_DIFF && diff <= MAX_TICK_DIFF,
```

Recommendation

We recommending deleting the redundant code snippet.

Alleviation

The client heeded the advice and resolved this issue in the commit `92b2cfbdb42645f2ede93acbe754ed1020f0ad13`.

LSS-02 | Lack of Safety Check

Category	Severity	Location	Status
Logical Issue	Minor	LixirStrategySimpleGWAP.sol: 26	Acknowledged

Description

The function `LixirStrategySimpleGWAP.initializeVault()` can be called by the `factory_role` to initialize the vault contract. However, this function can also be called after the initialization is done without any restriction, which may bring unexpected hazardous consequences.

Recommendation

We recommend adding necessary precautions to check the establishment of the vault contract.

Alleviation

[The Lixir Team]: The same logic can be performed with `configureVault` by the strategist, so a guard here wouldn't solve this problem. Being able to call `configureVault` again is intended, such as in cases of changing strategies or atomically changing multiple parameters.

LSS-03 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirStrategySimpleGWAP.sol: 61, 70, 83~85, 121~124	Acknowledged

Description

The `require` statements on the aforementioned lines do not have proper error messages implemented.

Recommendation

We recommend adding proper error messages to the `require` statements on the aforementioned lines.

Alleviation

[The Lixir Team]: This will only be done when a privileged account creates a new vault, we can debug manually.

LSS-04 | Missing Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	Informational	LixirStrategySimpleGWAP.sol: 56, 65, 74	Partially Resolved

Description

The functions that affect the status of sensitive variables should be able to emit events as notifications to the users. For example,

- `LixirStrategySimpleGWAP.setTickShortDuration()` to update the variable `TICK_SHORT_DURATION`;
- `LixirStrategySimpleGWAP.setMaxTickDiff()` to update the max tick difference;
- `LixirStrategySimpleGWAP.setSpreads()` to update the `mainSpread` and `rangeSpread`;

Recommendation

We recommend emitting events for all the essential state variables that are possible to be changed during the runtime.

Alleviation

The client heeded our advice and resolved the issue by adding event emissions in the functions at the aforementioned lines. The changes are reflected in the commit `92b2cfbdb42645f2ede93acbe754ed1020f0ad13`.

LSS-05 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	LixirStrategySimpleGWAP.sol: 56, 65, 74, 91, 258	Acknowledged

Description

The `strategist` is an important role in the contract `LixirStrategySimpleGWAP`. The `strategist can operate on the following functions:

- `LixirStrategySimpleGWAP.setTickShortDuration()` to update the short duration for checking the recent time weighted tick average;
- `LixirStrategySimpleGWAP.setMaxTickDif()` to update the maximum tolerable difference between ticks;
- `LixirStrategySimpleGWAP.setSpreads()` to update the spreads of the main order and the range order;
- `LixirStrategySimpleGWAP.configureVault()` to update the vault configurations;
- `LixirStrategySimpleGWAP.rebalance()` to rebalance the concentration orders.

Recommendation

We recommend the client carefully manage the project's private keys and avoid any potential risks of being hacked. We also advise the client to adopt the `Timelock` contract with a reasonable delay to allow users to withdraw their funds, Multisig with community-selected 3-party independent co-signer, and/or DAO with transparent governance with the project's community in the project to manage the sensitive role accesses.

Alleviation

[The Lixir Team]: Duly noted - we are exercising good opsec on these fronts. Also, the damage that can be done is very minimal, only forcing a capital inefficient position.

LSS-06 | Reentrancy Attack Risks

Category	Severity	Location	Status
Logical Issue	Minor	LixirStrategySimpleGWAP.sol: 91, 26	Acknowledged

Description

The function `LixirStrategySimpleGWAP.initializeVault()` calls function `LixirStrategySimpleGWAP._configureVault()`:

```
39      _configureVault(
40        _vault,
41        fee,
42        TICK_SHORT_DURATION,
43        MAX_TICK_DIFF,
44        mainSpread,
45        rangeSpread
46      );
```

Then, the function `LixirStrategySimpleGWAP._configureVault()` calls the function `LixirStrategySimpleGWAP._rebalance()` on line 148. However, the function `LixirStrategySimpleGWAP._rebalance()` includes an external call:

```
307     vault.rebalance(mlower, mupper, rlower0, rupper0, rlower1, rupper1, fee);
```

After the execution of this external call, there are state variables written:

```
52      vaultData.timestamp = uint32(block.timestamp);
53      vaultData.tickCumulative = ticksCumulative[0];
```

Thus, the function `LixirStrategySimpleGWAP.initializeVault()` might be vulnerable to reentrancy attacks.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[The Lixir Team]: We will only be calling Uniswap V3 pools. We won't be dealing with ERC777 or other potentially malicious tokens. Put the call at the end, anyway.

LVE-01 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirVaultETH.sol: 40, 282	Acknowledged

Description

The `require` statements on the aforementioned lines do not have proper error messages implemented.

Recommendation

We recommend adding proper error messages to the `require` statements on the aforementioned lines.

Alleviation

[The Lixir Team]: This will only be done when a privileged account creates a new vault, and we can debug manually.

LVE-02 | Reentrancy Attack Risks

Category	Severity	Location	Status
Logical Issue	Minor	LixirVaultETH.sol: 87, 46	Acknowledged

Description

The function `LixirVaultETH.deposit()` calls the function `LixirVaultETH._depositETH()`. Then, the function `LixirVaultETH._depositETH()` might execute the following external calls:

```
129         TransferHelper.safeTransferFrom(
130             address(token1),
131             msg.sender,
132             address(this),
133             amount1In
134           );
```

```
138         TransferHelper.safeTransferFrom(
139             address(token0),
140             msg.sender,
141             address(this),
142             amount0In
143           );
```

```
146         weth9.deposit{value: amount1In}();
```

After the aforementioned executions, `LixirVault._depositStepTwo()` that has event emissions was called. Thus the function `LixirVault.depositETH()` might be vulnerable to reentrancy attacks.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[The Lixir Team]: Function `LixirVaultETH._depositStepTwo()` also calls mint on the vaults. We won't be dealing with ERC777 or other potentially malicious tokens.

LVL-01 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	LixirVault.sol: 452, 456, 483, 492, 501, 510, 514	Partially Resolved

Description

Certain accounts or addresses are important roles in the contract `LixirVault`. These addresses can operate on the following functions:

- `LixirVault.setMaxSupply()` to update the maximum supply of shares available to users;
- `LixirVault.setPerformanceFee()` to update the applicable fee ratio;
- `LixirVault.setKeeper()` to update the keeper address of the vault contract;
- `LixirVault.setStrategist()` to update the strategist address of the vault contract;
- `LixirVault.emergencyExit()` to withdraw all the tokens from Uniswap and pause the vault contract;
- `LixirVault.unpause()` to unpause the vault;
- `LixirVault.rebalance()` to rebalance the concentration orders.

Recommendation

We recommend the client carefully manage the project's private key and avoid any potential risks of being hacked. We also advise the client to adopt the `Timelock` contract with a reasonable delay to allow users to withdraw their funds, Multisig with community-selected 3-party independent co-signer, and/or DAO with transparent governance with the project's community in the project to manage the sensitive role accesses.

Alleviation

[The Lixir Team]: Duly noted, we are exercising good opsec on these fronts. Also, the damage that can be done is very minimal, only forcing a capital inefficient position.

LVL-02 | Unhandled Return Values

Category	Severity	Location	Status
Coding Style	Informational	LixirVault.sol: 689, 699, 690, 700, 746, 580, 622, 225, 235	Pending

Description

The functions `UniswapV3Pool.burn()`, `UniswapV3Pool.collect()` and `UniswapV3Pool.mint()` are not void-returning functions. Ignoring their return values, especially when their return values might represent the status if the transaction is executed successfully, might cause unexpected exceptions.

Recommendation

We recommend handling the return values of functions `UniswapV3Pool.burn()`, `UniswapV3Pool.collect()` and `UniswapV3Pool.mint()` before continuing processing.

Alleviation

[The Lixir Team]: These will always revert if the internal transaction fails. We don't need the return values, as the calculations are ensured elsewhere.

LVL-03 | Redundant Contract Import

Category	Severity	Location	Status
Coding Style	Informational	LixirVault.sol: 15~16	Resolved

Description

The contract `Pausable.sol` is imported twice.

Recommendation

We recommend removing the redundant contract import.

Alleviation

The client heeded our advice and resolved the issue by removing the redundant import. The change is reflected in the commit `92b2cfbdb42645f2ede93acbe754ed1020f0ad13`.

LVL-04 | Missing Event Emissions for Significant Transactions

Category	Severity	Location	Status
Coding Style	Informational	LixirVault.sol: 452, 456, 483, 492, 501, 510	Acknowledged

Description

The functions that affect the status of sensitive variables should be able to emit events as notifications to the users. For example,

- `LixirVault.setMaxSupply()` to update the maximum supply;
- `LixirVault.setPerformanceFee()` to update the fee rate;
- `LixirVault.setKeeper()` to update the keeper role;
- `LixirVault.setStrategist()` to update the stragegist role;
- `LixirVault.emergencyExit()` to pause the vault contract and emergency exit;
- `LixirVault.unpause()` to unpause the vault contract.

Recommendation

We recommend emitting events for all the essential state variables that are possible to be changed during the runtime.

Alleviation

[The Lixir Team]: We simply cannot afford more bytecode due to bytecode size limit. Contract `Pausable` from OZ already implements events.

LVL-05 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	LixirVault.sol: 127, 137, 142, 147, 460, 479, 523~531, 1275	Acknowledged

Description

The `require` statements on the aforementioned lines do not have proper error messages implemented.

Recommendation

We advise the client to add proper error messages to the `require` statements on the aforementioned lines.

Alleviation

N/A

LVL-06 | Incompatibility with Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Informational	LixirVault.sol: 254, 386, 408	Acknowledged

Description

The contracts `LixirVault` and `LixirVaultETH` operate as the main entries for the interaction with the users. The users deposit token pairs and store them in a vault, and the token pairs are provided as liquidity using Uniswap V3. Later on, the users can withdraw their assets from the vault. In this process, `deposit()`, `withdraw()`, or `withdrawFrom()` may be involved in transferring users' assets into or out of the Lixir protocol. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistency.

Recommendation

We recommend keeping regulating the set of token pairs supported by the Lixir Protocol, and if there is a need to support deflationary tokens, add necessary mitigation mechanisms to keep track of accurate balances.

Alleviation

[The Lixir Team]: We don't plan to support deflationary tokens at the moment. If we ever do, we will make appropriate changes with additional audits.

LVL-07 | Reentrancy Attack Risks

Category	Severity	Location	Status
Logical Issue	Informational	LixirVault.sol: 386, 408, 501, 514, 558, 254	Pending

Description

The aforementioned functions have external calls before state variable changes or event emissions. Thus these functions are vulnerable to reentrancy attacks.

Recommendation

We recommend applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

LVL-08 | Inaccurate Calculation

Category	Severity	Location	Status
Logical Issue	Medium	LixirVault.sol: 792~793, 798, 808, 800, 809	Resolved

Description

The function `LixirVault.calcSharesAndAmounts()` calculates the shares to be granted and the amounts to deposit based on the amounts desired by the users.

- 1. In the following code snippet, two variables are introduced to offset the influence brought by rounding up. However, the offsets are `1` and `2`, instead of `0` and `1`. In addition, the offsets are larger for the share that is not rounded up than the one rounded up, which is counterintuitive.

```
792     uint8 realSharesOffsetFor0 = roundedSharesFrom0 ? 1 : 2;
793     uint8 realSharesOffsetFor1 = roundedSharesFrom1 ? 1 : 2;
```

- 2. The amount of shares granted to the users is defined as the remainder of `sharesFrom0` subtracting `realSharesOffsetFor0` and `1`. If it is in a rounded-up case, the amount of shares would be `sharesFrom0 - 1 - 1 = sharesFrom0 - 2`, which is less than the number of shares the users should acquire by `1`. Similarly, in a not-rounded-up case, `shares = sharesFrom0 - 3`, which indicates that the amount of shares granted to users is less than the amount that ought to be by `3`.

```
798       shares = sharesFrom0 - 1 - realSharesOffsetFor0;
```

```
808       shares = sharesFrom1 - 1 - realSharesOffsetFor1;
```

- 3. After the amount of shares has been calculated, the `amount0In` and `amoount1In` should be derived from the `shares`. However, the `amount0In` and `amount1In` are calculated based on the rounded-up amount `sharesFrom0` and `sharesFrom1`, as shown in the code snippets below.

```
800         amount1In = FullMath.mulDivRoundingUp(sharesFrom0, total1, _totalSupply);
```

```
809         amount0In = FullMath.mulDivRoundingUp(sharesFrom1, total0, _totalSupply);
```

Do you mind explaining the reason behind this implementation to us?

Alleviation

[The Lixir Team]: The 1 and 2 is actually by design. The purpose here is twofold: one, we want to have all rounding errors to be in the houses favor, at the expense of users, to prevent rounding errors letting a deposit and immediate withdrawal result in a profit. Hence, shares are rounded down by 2, or 3 if the roundedUpShares resulted in a rounding value. However, for amounts in, we use the rounded up value. The other reason for this is so that we definitely have enough to deposit in appropriate proportions in the main and range orders, in fuzzing tests, without this overestimation, sometimes we'd be short by a few wei without this aggressive rounding up.

The implementation here is using Uniswap V3's mulDiv math, which multiplies two 256 bit integers into a two 256 bit word (512 bit total) value, and then divides by the divisor. our internal mulDivRoundingUp is a straightforward modification of their mulDivRoundingUp to signal whether or not we rounded, this is the for the purpose of the "rounding in the house's favor" calculation of shares.

LVL-09 | Possibly Uninitialized Variables

Category	Severity	Location	Status
Logical Issue	Minor	LixirVault.sol: 558	Resolved

Description

The variables `LixirVault.mintPosistion().rangeData` and `LixirVault.mintPosistion().rL` might be uninitialized if `rL1 = rL2 = 0`. We understand that this case would be extremely rare. However, we want to inform the team in case that the aforementioned scenario happens and causes unexpected problems.

Recommendation

We recommend handling the aforementioned case to avoid unexpected problems.

Alleviation

[The Lixir Team]: The rebalance function resets these from values passed from the strategy. For deposits, rL will return 0 from _calculateTotals. UniswapV3Pool.positions is a simple mapping getter, and will return all 0's if it's an invalid position such as { tickLower: 0, tickUpper: 0}.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.