LevelQ
Audit Report

Wed Jun 18 2025

contact@bitslab.xyz https://twitter.com/tonbit_

TonBit

LevelQ Audit Report

1 Executive Summary

1.1 Project Information

Description	LevelQ addresses the complexities within the TON ecosystem by offering a streamlined approach to DeFi, designed to make it easier for users to discover, access, and optimize their digital assets.
Type	DeFi
Auditors	TonBit
Timeline	Mon May 19 2025 - Wed Jun 18 2025
Languages	FunC
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/LevelQ-NMC/bonus-liquidity-contract
Commits	c129c87306b6b8d5d7d03ceebcf2e120fe0af354 680e7e1318d807fad6e26c8ad7d08df878e1d5b7 f18aa7938b769fd640639dc21b601a1b05341eca

1.2 Files in Scope

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

ID	File	SHA-1 Hash
AMM	contracts/lp_contract/amm.fc	7d91a5ab8a5b131bfc44d9b44476ab39ee24dad5
ERR	contracts/lp_contract/errors.fc	101c70d321ab0a2d9e8f17470cad560e4da86254
OP	contracts/lp_contract/op.fc	ce05175bb7d3657f10cc835cf9bf37801b249917
STO	contracts/lp_contract/storage.fc	66bf337cd57ad7236a48f74d558520b1354ed6f1
LAU	contracts/lp_contract/lp_account-utils.fc	47bb4332c3baf5eac647c6962850019fd07def10
SRE	contracts/lp_contract/roles/sudoer_requests.fc	145b32dddfb205a833059f17c6c4930ed38bfd80
HRE	contracts/lp_contract/roles/halter_requests.fc	4234f9700809761ca8f1edf8d71028091b8c963a
GRE	contracts/lp_contract/roles/governor_requests.fc	feca15e1fd4daa76a3f6d758e258ee4dc36e5a98
PAR	contracts/lp_contract/params.fc	06b39d83f2cd5d12351234832a29f89749f97dc5
GET	contracts/lp_contract/get.fc	8eb934895a679964c19bf6567620275e9408e36c

ASS	contracts/lp_contract/asserts.fc	80c2152fee28b8b4f3cef1f369a2cde23b70d04e
ERR1	contracts/lp_account/errors.fc	3d4932214719cd033a065888b1de414b718db649
OP1	contracts/lp_account/op.fc	59cec21b49338dd754a9e40a800da1a833f81e34
STO1	contracts/lp_account/storage.fc	2922cee7c447208aa6dc2ece7334aecaa47ba1e7
PAR1	contracts/lp_account/params.fc	1990d10a579a39a3f35b2e6b64ac59c96cdd22d3
PCA	contracts/lp_account/pool-calls.fc	d6444fa1d015d9ac2f71fdffb9a27e951c9ee037
GET1	contracts/lp_account/get.fc	8ac3d9b371fcef2be9e96c9269f8baa2a13353c1
LAC	contracts/lp_account.fc	6c67bad342a5e759a62d96ebfd23f19f3c71b063
LCO	contracts/lp_contract.fc	73a64cb679d0e922c54ef939765fc5fddb5c4a57

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	9	2
Informational	5	4	1
Minor	1	1	0
Medium	2	2	0
Major	1	0	1
Critical	2	2	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Sachin to identify any potential issues and vulnerabilities in the source code of the LevelQ smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 11 issues of varying severity, listed below.

ID	Title	Severity	Status
ERR-1	Unused Fields	Informational	Fixed
HRE-1	Inconsistent Roles for Halting and Unhalting the Contract	Informational	Fixed
LCO-1	Malicious Attacker Can Forge `pool::cb_deposit` to Drain Pool Funds	Critical	Fixed
LCO-2	Parameter Verification is Incomplete	Critical	Fixed
LCO-3	Centralization Risk	Major	Acknowledged
LCO-4	Halt Mechanism May Cause User Fund Loss	Medium	Fixed
LCO-5	Price Time Validity Mechanism May Cause User Fund Loss	Medium	Fixed
LCO-6	Wrong Comment	Informational	Fixed

LCO-7	Unused Storage Variable `storage::bonus_fee`	Informational	Acknowledged
PCA-1	Ratio Sum Should Be 100	Minor	Fixed
STO-1	Mismatch in Storage and Transmission Bitwidth for Ratio Values	Informational	Fixed

3 Participant Process

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

halter

- Can halt deposit and withdraw

sudoer

- Can send any message in the role of pool
- Can update pool's storage

governor

- Can update sudoer,governor,interest manager,halter
- Can set unhalt deposit and withdraw

interest manager (Oracle)

- Can set LP price
- Can set bonus fee

4 Findings

ERR-1 Unused Fields

Severity: Informational

Status: Fixed

Code Location:

contracts/lp_account/errors.fc#1;

contracts/lp_contract/params.fc#1;

contracts/lp_account/errors.fc#1;

contracts/lp_account/params.fc#1

Descriptions:

1. In the `lp_contract` folder, the `error::wrong_state` and `NO_LIQUIDITY` fields in the `errors.fc` file are not used,
2. and the `FEE_DIVIDER` and `TON_ADDRESS` fields in the `params.fc` file are not used;
3. in the `lp_account` folder, the `NO_LIQUIDITY` and `proportion::wrong_ratio` fields in the `errors.fc` file are not used,
4. and the `REQUIRED_TON_RESERVE` , `FEE_DIVIDER` , and `TON_ADDRESS` fields in the `params.fc` file are not used.

Suggestion:

It is recommended to remove unused fields.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

HRE-1 Inconsistent Roles for Halting and Unhalting the Contract

Severity: Informational

Status: Fixed

Code Location:

contracts/lp_contract/roles/halter_requests.fc#2;

contracts/lp_contract/roles/governor_requests.fc#8

Descriptions:

The `halter` role can pause the contract, while unpausing requires the `governor` role.

Typically, both operations should be managed by the same role. See code below:

```
() process_halt_request(slice sender) impure inline_ref {
    assert_sender(sender, storage::halter_address);
    storage::halted = true;
}
() process_unhalt_request(slice sender) impure inline_ref {
    assert_sender(sender, storage::governor_address);
    storage::halted = false;
}
```

Suggestion:

Use the same role for both pausing and unpausing the contract.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-1 Malicious Attacker Can Forge `pool::cb_deposit` to Drain Pool Funds

Severity: Critical

Status: Fixed

Code Location:

contracts/lp_contract.fc#42-82

Descriptions:

The `lp_contract.fc` contract does not verify whether the `pool::cb_deposit` message is sent by an authentic `stonfi wallet` or `dedust wallet`, allowing any contract to forge the message.

```
() process_jettons(slice in_msg_body, slice sender_address) impure {
    ...
    if (op == pool::cb_deposit){
        slice user_lp_address_generated = calculate_user_lp_account_address(...);
        throw_unless(error::invalid_caller, equal_slices(user_lp_account,
user_lp_address_generated));
```

In the above `process_jettons()` function, the argument of `sender_address` (i.e., stonfi or dedust wallet) is not validated, allowing an attacker to fake a `pool::cb_deposit` message and drain the pool.

Suggestion:

Add a check to ensure `sender_address` is a valid `stonfi` or `dedust` LP wallet.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-2 Parameter Verification is Incomplete

Severity: Critical

Status: Fixed

Code Location:

contracts/lp_contract.fc#155-165

Descriptions:

If the message is not sent from the `lp_account` contract, there might be a case of message forgery. An attacker could forge `user_address` and `user_lp_account` addresses to bypass this check: `throw_unless(error::invalid_caller, equal_slices(user_lp_account, user_lp_address_generated));` Once successfully bypassing the check, the attacker can forge arbitrary amounts of `stonfi` and `dedust` tokens to stake for acquiring `tby` tokens, leading to significant security risks and asset losses.

Suggestion:

Pass the `from_address` into the `process_jettons` function, and during the check `throw_unless(error::invalid_caller, equal_slices(user_lp_account, user_lp_address_generated));` inside the function, instead of using the `user_lp_account` parsed from the `in_msg_body`, directly use the passed-in `from_address`.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-3 Centralization Risk

Severity: Major

Status: Acknowledged

Code Location:

contracts/lp_contract.fc#106

Descriptions:

The `sudoer` role in `lp_contract.fc` can send arbitrary messages as the pool and modify any pool storage.

The `governor` can assign the `sudoer`.

Suggestion:

Use multisig for both `sudoer` and `governor`.

Resolution:

The team acknowledges the centralisation of sudoer and governor and will transition to multisig soon.

LCO-4 Halt Mechanism May Cause User Fund Loss

Severity: Medium

Status: Fixed

Code Location:

contracts/lp_contract.fc#156

Descriptions:

When a `pool::cb_deposit` is received while the contract is halted, it won't mint `tby` tokens for users, or return the received assets.

```
} elseif (op == jetton::transfer_notification) {
  assert_not_halted();
  ...
  process_jettons(cs.begin_parse(), sender_address);
```

Suggestion:

The halt mechanism should not affect `pool::cb_deposit` handling.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-5 Price Time Validity Mechanism May Cause User Fund Loss

Severity: Medium

Status: Fixed

Code Location:

contracts/lp_contract.fc#57

Descriptions:

If the price set by oracle is older than 30 minutes when a `pool::cb_deposit` is received, the pool won't mint `tby` tokens for users, or return the received assets.

```
if ((storage::lp_price_updated_at + 1800 < now()) & (storage::tby_lp_token_supply > 0)){
  throw(error::lp_price_outdated);
```

Suggestion:

The price time validity check should not apply to `pool::cb_deposit`.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-6 Wrong Comment

Severity: Informational

Status: Fixed

Code Location:

contracts/lp_contract.fc#184-186

Descriptions:

```
if ((storage::stonfi_lp_ratio == 0) | (storage::dedust_lp_ratio == 0)){
        required_ton = ONE_TON * 5 / 3; ;; 1.25 TON needed, most of them will be
returned
    }
```

The code and comments do not correspond, 1.25 is 5/4.

Suggestion:

It is recommended to keep code and comments consistent.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LCO-7 Unused Storage Variable `storage::bonus_fee`

Severity: Informational

Status: Acknowledged

Code Location:

contracts/lp_contract.fc#135

Descriptions:

The `interest manager` role sets `storage::bonus_fee`, but it's never used elsewhere.

```
} elseif (op == interest_manager::set_interest){
    assert_sender(...);
    storage::bonus_fee = in_msg_body~load_uint(8);
```

Suggestion:

Consider removing the unused storage.

Resolution:

Can be reused in future

PCA-1 Ratio Sum Should Be 100

Severity: Minor

Status: Fixed

Code Location:

contracts/lp_account/pool-calls.fc#16

Descriptions:

The following code lacks a check enforcing `stonfi_lp_ratio + dedust_lp_ratio == 100`, despite the comment:

```
() handle_pool_messages(...) impure inline {
  if (op == pool::deposit_ratio){
    (int stonfi_lp_ratio, int dedust_lp_ratio) = (...); ;; ratio sum should be 100
```

Suggestion:

Add a check to enforce the ratio sum equals 100.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

STO-1 Mismatch in Storage and Transmission Bitwidth for Ratio Values

Severity: Informational

Status: Fixed

Code Location:

contracts/lp_contract/storage.fc#77

Descriptions:

In `contracts/lp_contract/storage.fc` :

```
storage::stonfi_lp_ratio = ds~load_uint(16);
storage::dedust_lp_ratio = ds~load_uint(16);
```

But when transmitted (in `contracts/lp_contract.fc`):

```
cell payload = begin_cell()
  .store_uint(..., 32)
  ...
  .store_uint(storage::stonfi_lp_ratio, 8)
  .store_uint(storage::dedust_lp_ratio, 8)
  .end_cell();
```

The values are stored as `uint16` but sent as `uint8` . If the ratio exceeds `uint8` max, it breaks functionality.

Suggestion:

Unify storage and transmission bitwidth.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.