HyperPool Quadrat Security Analysis

by Pessimistic

This report is public

March 21, 2023

Abstract

In this report, we consider the security of smart contracts of HyperPool Quadrat project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. A single audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Moreover, we recommend auditing the whole repository. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of HyperPool Quadrat smart contracts. We described the audit process in the section below.

The initial audit showed two critical issues that allowed to steal the funds: C1, C2. The audit also revealed several issues of medium severity: Delegatecall to an arbitrary address, Pool manager privileges, Slippage problems, No tests, and Factory manager role. Moreover, several low-severity issues were found.

Please note that the scope of work included only parts of the project (see the Audit process section).

After the initial audit, the developers provided several versions of the code. The security report represents the results for the latest version we have observed. The list of all versions can be found in the Codebase update section. As a result, the developers fixed all issues of critical severity and either fixed or commented on most of the other issues. In addition to this, the developers added tests.

Two issues of low severity were removed from the report. L02 issue was identified as a false positive case, and L06 issue was related to the contract that is out of the scope of the works. M06 issue was found during the recheck process and was not present in the initial report.

General recommendations

We recommend conducting an audit of the whole project. In order to understand what was audited as a part of this security review, see the Audit process section.

Project overview

Project description

For the audit, we were provided with a private repository of the **HyperPool Quadrat** project.

The scope of the audit includes:

- Difference between commits `0a86e86fd6c2c77ba445c7c16f2f7be7e5cdfc30` and `26c9d2f4e06f6ed8b36d698916d7332ee180bacd`;
- **HyperLPRouter.sol** file;
- **HyperLibrary.sol** file.

The documentation for the project includes **README.md** file in the repository and the following link.

The project does not contain tests.

Codebase update

After the initial audit, the developers provided several versions of the code in the following zip archives:

- **hyperpools_calldata_10.03.2023.zip**, sha1sum **953e79faad402229af28d670ee01548c2a1ceb77**;
- **hyperpools_call_data_20.03.2023.zip**, sha1sum **615e3a01a3c1145d96e64bf7c298a6293b735352**;
- **hyperpools_call_data_20.03.2023_upd.zip**, sha1sum **d7bfd66e36a7ab0f886a940a1fdc984e9b9eeaff**.

As a result, the latest version does not contain critical issues. Moreover, the developers fixed and commented on several other issues.

The new version of the code contains tests. 21/21 tests pass, and coverage is 69.2%.

Audit process

We started the audit on February 22 and finished on March 6, 2023.

For the audit, we checked the difference in the code between the following commits: `0a86e86fd6c2c77ba445c7c16f2f7be7e5cdfc30` and `26c9d2f4e06f6ed8b36d698916d7332ee180bacd` (see the Project description section). We inspected the provided materials and the documentation and conducted a call with the developers. The developers instructed us to check the new functionality and the modified one, considering the previous version of the code secure. The developers asked to pay attention to the following functions:

- `mintHyper` function in the **HyperLPool.sol** file;
- `rebalance` function in the **HyperLPool.sol** file;
- `executiveRebalace` function in the **HyperLPool.sol** file.

After the investigation, we conducted that for the audit we need to fully audit the following files:

- **HyperLPRouter.sol**;
- **HyperLibrary.sol**.

In addition to this, we were required to investigate the rest of the project even if the logic was not changed.

Since the **HyperLPool** contains an arbitrary call to the hyperdex router, which is out of the scope, we investigated the possible called logic of that contract. That router uses 0x project for the swaps. In order to find possible attack vectors, we dived into the 0x documentation.

The project utilizes several roles. According to the developers, the manager of the pool is not a trusted role, while others are trusted ones.

Due to the aforementioned limitations of this audit, we assumed that the following parts of the codebase were secure and did not check them thoroughly:

- The mathematics of the pools that has not been changed. Note that this logic was tested in the previous versions of the code, while the latest version of the code does not contain tests (see M04 issue);
- Code logic that was implemented prior to the commit `26c9d2f4e06f6ed8b36d698916d7332ee180bacd`;
- We considered the factory manager role as a trusted one. We covered the privileges of this role and associated risks in M01 and M05 issues;
- External projects such as 0x and Uniswap V3.

Note that we did not audit the whole hyperdex router since it is out of the scope. However, we checked how it could be utilized for the attacks on the project.

We scanned the project with the static analyzer Slither and our private plugin with an extended set of rules and then manually verified their output.

We combined in a private report all the verified issues we found during the manual audit or discovered by automated tools.

On March 10, the developers provided a new version of the code. We reviewed the fixes for the issues and inserted comments from the developers. As a result, we contacted the developers regarding several fixes and concluded that the provided version was not optimal for the production environment. After that, we checked two more versions of the project.

We gathered all the information in the report based on the latest provided version.

Manual analysis

The contracts were completely manually analyzed, their logic was checked. Besides, the results of the automated analysis were manually verified. All the confirmed issues are described below.

Critical issues

Critical issues seriously endanger project security. They can lead to loss of funds or other catastrophic consequences. The contracts should not be deployed before these issues are fixed.

C01. **Stolen funds** (fixed)

A malicious user can steal tokens from users who approved their tokens to **HyperLPRouter** contract with the following attack:

1. An attacker deploys a malicious token and creates a Uniswap pool and a HyperPool pool. The second token in a pair is the token that attacker wants to steal.
2. The attacker calls `HyperLPRouter.mint` function.
3. `mint` function calls internal `_swap` that contains `swapCall` modifier. As a result, the transaction is in `_IN_SWAP` state.
4. In `_swap` function `IUniswapV3Pool(pool).swap` function is called, which calls `safeTransfer` of the malicious token.
5. The malicious token calls `HyperLPRouter.uniswapV3SwapCallback`, providing the token address as `pool` and the victim user address as `payer`. Notice that this function will succeed since the router contract is in `_IN_SWAP` state, and the check at line 354 also passes. As a result, tokens from the `payer` user are stolen.

The issue has been fixed and is not present in the latest version of the code.

C02. **Stolen funds** (fixed)

HyperLPool contract contains `executiveRebalance` function. This function allows the manager to rebalance funds. In order to do so, these funds need to be swapped. Normally, the manager provides `calldata` for the hyperdex router. In the current implementation, the router calls 0x router, which swaps tokens. However, there is no check for the function called at the 0x router. Hence, the manager is able to steal funds by filling 0x order instead of swapping funds. We recommend adding additional checks to the used `calldata` in addition to the router address check.

The issue has been fixed and is not present in the latest version of the code.

Medium severity issues

Medium severity issues can influence project operation in the current implementation. Bugs, loss of potential income, and other non-critical failures fall into this category, as well as potential problems related to incorrect system management. We highly recommend addressing them.

M01. Delegatecall to an arbitrary address (addressed)

Line 94 in **Proxy.sol** file contains `delegatecall` to a new implementation. Despite the fact that this function is called by a trusted role, this line allows a malicious actor to delete the whole contract by making `delegatecall` to a contract with `selfdestruct` instruction. We recommend carefully maintaining a list of trusted addresses.

The developers decided to use multisig wallet for the factory manager address.

M02. Pool manager privileges (commented)

The pool manager is able to change several crucial parameters of the pool: `managerFeeBPS`, `hyperpoolsRebalanceBPS`. These parameters change the applied fees in the pool. We recommend either retracting these privileges or implementing additional checks for the possible values.

According to the developers, this is a part of a business logic. It is safe since the commissions are only taken from the pool profit and the manager has the right to spend them at their own discretion.

M03. Slippage problems (commented)

In the code, 0x router is used for the swap process for the pool rebalancing. Note, that the allowed slippage is provided by the manager. Since there are no checks for the slippage parameter, this opens the possibility for the flashloan attack on the pool. We recommend considering using TWAP for the prices logic.

According to the developers, this is an economic vulnerability that cannot be addressed. Moreover, the developers pointed out that they will notify the users about it.

M04. No tests (fixed)

The project does not contain tests. Testing is important for the project security and to mitigate possible bugs. Hence, we recommend covering the code with tests.

The issue has been fixed and is not present in the latest version of the code.

M05. **Factory manager role** (addressed)

The `manager` role of **HyperLPFactory** contract is crucial for the correct code operation. It has the following powers:

- It can upgrade the logic of the factory contract and existing pools;
- Make pools immutable;
- Upgrade the new pools implementation logic;
- Mark pools as trusted and untrusted;
- Set `hyperpoolsDeployer` address.

There are scenarios that can lead to undesirable consequences for the project and its users, e.g., if private keys for the role become compromised. We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

The developers decided to use multisig wallet for the factory manager address.

M06. **Missing approves** (fixed)

In `_swapAndDeposit` function of **HyperLPool** contract, and `mintHyper` function of **HyperLPRouter** contract, **Hyperdex** is used for swaps. But the contracts are not giving approval for the tokens, which means swaps will fail. We recommend giving approval for the tokens that are used in the swap.

The issues have been fixed and are not present in the latest version of the code.

Low severity issues

Low severity issues do not directly affect project operation. However, they might lead to various problems in future versions of the code. We recommend fixing them or explaining why the team has chosen a particular option.

L01. No lock file (fixed)

The project does not contain `yarn.lock` file. We recommend adding it in order to make the development process easier when the project is maintained by multiple developers in the project.

The issue has been fixed and is not present in the latest version of the code.

L03. Differing constants for ETH (fixed)

The project uses a hardcoded address for the native currency. However, used hardcoded address is different in **Hyperfied.sol** and **HyperLPRouter.sol** files. We recommend using the same address for similar constants.

The issue has been fixed and is not present in the latest version of the code.

L04. Gas optimization

It is safe to give infinite approval in **HyperLPRouter** router since the router should be stateless. This allows saving gas.

L05. Initialization of implementation contracts

The common pattern for upgradeable contracts is to move all the contract initialization into `initialize` function. This allows to initialize proxy storage. However, there is an attack vector that exploits `initialize` function by making a call directly to the logic contract. The current implementation is safe from this type of attack. Nevertheless, we recommend preventing logic contract initialization in order to make future versions of the code more secure.

*The developers added `_disableInitializers()` in the constructor of **HyperLPoolStorage**, but we also recommend adding this in **HyperLPFactoryStorage** contract.*

L07. Optimization (commented)

In **hardhat.config.ts** file, the developers use optimization value `1` as `runs`. Setting `runs` as `1` will produce the code that is optimized only for the deployment. Given the fact that the contracts are intended to be used after the deployment, consider increasing the value of `runs` to make the compiler optimize the code to decrease the execution costs in the project.

The contracts cannot be more optimized since future optimizations will produce the code that is too large for the deployment.

L08. Redundant code (commented)

HyperLPoolStorage contract contains redundant variables that are not used: `hyperpoolsSlippageInterval` and `hyperpoolsSlippageBPS`. These variables are in the project due to upgradeability logic: they were used in previous versions of the code. However, since they are not used in the latest implementations, we recommend not setting them in the `initialize` function in order to save gas.

The developers intentionally decided to retain the mentioned code.

L09. Redundant dependencies (fixed)

The code contains redundant imports that are not used in the project:

- `IUniswapV3SwapCallback` import in **HyperLPool.sol** file;
- `TYPES` import in **HyperLPRouter.sol** file.

We recommend removing them.

The issue has been fixed and is not present in the latest version of the code.

L10. Unfinished code (fixed)

HyperLibrary.sol file in the development since it contains TODO comments and commented blocks of the code.

The issue has been fixed and is not present in the latest version of the code.

This analysis was performed by Pessimistic:

Pavel Kondratenkov, Security Engineer
Yhtyyar Sahatov, Junior Security Engineer
Irina Vikhareva, Project Manager
Alexander Seleznev, Founder

March 21, 2023