



VirtuSwap Security Analysis

by Pessimistic

This report is public

July 25, 2022

Abstract	2
Disclaimer	2
Summary	2
General recommendations	2
Project overview	3
Project description	3
Codebase update #1	3
Procedure	4
Manual analysis	5
Critical issues	5
C01. Incorrect LP token quantity calculation (fixed)	5
Medium severity issues	6
M01. Discrepancy with the documentation (fixed)	6
M02. No tests for the project (fixed)	6
M03. Incorrect LP token calculation (fixed)	6
M04. Incorrect calculation of the required amount for a given token (fixed)	6
M05. Incorrect parameters order (fixed)	7
M06. Overpowered role (new)	7
Low severity issues	8
L01. Unnecessary SafeMath usage (fixed)	8
L02. Dependency management issues (fixed)	8
L03. Code quality - constants (fixed)	8
L04. Code quality - value (fixed)	8
L05. Code quality - calculations (fixed)	9
L06. Code quality - import (fixed)	9
L07. Code quality - redundant function (fixed)	9
L08. Code quality - redundant contract (fixed)	9
L09. Code quality - redundant check (fixed)	9
L10. Code quality - external (fixed)	9
L11. Code quality - dead code (fixed)	10
Notes	10
N01. Design drawback	10

Abstract

In this report, we consider the security of smart contracts of the [VirtuSwap](#) 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. Besides, a security audit is not investment advice.

Summary

In this report, we considered the security of [VirtuSwap](#) smart contracts. We performed our audit according to the [procedure](#) described below.

The audit showed one critical issue: [Incorrect LP token quantity calculation](#). The audit also revealed several issues of medium severity: [Discrepancy with the documentation](#), [No tests for the project](#), [Incorrect LP token calculation](#), [Incorrect calculation of the required amount for a given token](#), and [Incorrect parameters order](#). Moreover, several low-severity issues were found.

After the initial audit, the codebase was [updated](#).

In this update, the developers fixed the [Incorrect LP token quantity calculation](#) issue of critical severity and all issues of medium severity, including [Discrepancy with the documentation](#), [No tests for the project](#), [Incorrect LP token calculation](#), [Incorrect calculation of the required amount for a given token](#), [Incorrect parameters order](#), and all issues of low severity. However, we discovered one more issue [Overpowered role](#) and added the comment from the developers. Also, all tests passed and the code coverage was measured.

The code quality is good. The security of [UniswapV2](#) in this project is preserved.

General recommendations

We recommend improving NatSpec coverage. We also recommend implementing CI to run tests, and analyze code with linters and security tools.

Project overview

Project description

For the audit, we were provided with [VirtuSwap](#) project on a private GitHub repository, commit [3b9ad51f90e117599a73005192ecdd6ca02cd8af](#).

The scope of the audit includes everything.

The documentation for the project includes a [Notion file](#).

```
sha1sum - df199024dcfd4da9af0ea11986339255c1901b99.
```

The total LOC of audited sources is 944.

Codebase update #1

After the initial audit, we were provided with commit [7f30082ad657c70d8725b562624cf5354222708](#).

In this update, the developers fixed all issues and added new functionality. However, in the updated codebase, we discovered the issue of medium severity. Also, the developers added tests and all 39 tests passed. The code coverage was measured and increased 91.53%.

Procedure

In our audit, we consider the following crucial features of the code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices.

We perform our audit according to the following procedure:

- Automated analysis
 - We scan the project's codebase with the automated tool: [Slither](#).
 - We manually verify (reject or confirm) all the issues found by the tool.
- Manual audit
 - We manually analyze the codebase for security vulnerabilities.
 - We assess the overall project structure and quality.
- Report
 - We reflect all the gathered information in the report.

Inter alia, we check:

- Whether the code implements all the important checks that [UniswapV2](#) has in the corresponding contracts.
- The logic of LP tokens accrual, including the case when `totalSupply = 0`.
- Whether the code and the documentation are consistent.
- Standard Solidity issues in the codebase.
- The new logic that is associated with virtual pools.

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. Incorrect LP token quantity calculation (fixed)

While calling directly (not through **vRouter**) the `mint` function of the **vPair** contract, the quantity of LP tokens provided for liquidity is calculated using parameters `_reserve0` and `amount0` only for one token in pair. This may lead to an unexpectedly large amount of withdrawal of the other token in pair.

Consider checking how this vulnerability is prevented in the [UniswapV2Pair](#) contract where the minimum quantity of LP tokens in pair is taken.

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

Medium severity issues

Medium 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. Discrepancy with the documentation (fixed)

In the provided [documentation](#), flashswap functionality is mentioned. However, the stated functionality is not working due to the current balance check before the external call `vSwapcallee` in the `swapNative` function of the `vPair` contract.

Consider moving the check of the current balance after the external call.

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

M02. No tests for the project (fixed)

The project has no tests at all. We always note the availability of tests as well as code coverage. We highly recommend covering the code with tests and making sure that all tests pass and the code coverage is sufficient.

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

M03. Incorrect LP token calculation (fixed)

The intention of the `reserveRatio` variable is unclear. In the current implementation, the `reserveRatio` variable equals 0, and the amount of LP tokens is calculated regarding only the tokens of the pair. The reserve tokens are not included in the calculation. However, when LP tokens are burnt, the reserve tokens are included in the calculations, which means that they will be withdrawn from the pair.

Consider adding comments about this formula.

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

M04. Incorrect calculation of the required amount for a given token (fixed)

Functions `quoteInput` and `quoteOutput` of `vSwapMath` contract do not follow the rule $\text{quoteInput}(\text{quoteOutput}(x)) = x$ as it is done in [UniswapV2Library](#). Also, the fee is calculated incorrectly at lines 75 and 89. This leads to the incorrect operation of swaps and liquidity provision.

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

M05. Incorrect parameters order (fixed)

Parameters of the `calculateLPTokensAmount` function are passed in the wrong order in function `mint` of the **vPair** contract. The correct parameters order is:

1. `amount0`;
2. `totalSupply`;
3. `_reserve0`;
4. `reserveRatio`;

The `calculateLPTokensAmount` function has been removed and this issue is not present in the latest version of the code.

M06. Overpowered role (new)

In the **vPairFactory** and in the **vPair** contracts, admin can set and change:

1. the swap fees;
2. the max reserve ratio;
3. the max number of whitelisted tokens of a pair;
4. set whitelisted tokens;
5. change the address of the factory.

In the **vRouter** contract, owner can change the address of the factory.

If the admin's private keys become compromised, this could lead to the admin being able to front-run a transaction of the pair creation and set the parameters of a pair. Also, the admin can front-run the swap functions.

We recommend designing contracts in a trustless manner or implementing proper key management, e.g., setting up a multisig.

Comment from the developers: In the first version, admin keys to control the pool parameters will be held in a multisig wallet owned by Virtuswap foundation. The next version will introduce different features to move the protocol control to governance / DAO as described in our tokenomics.

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. Unnecessary SafeMath usage (fixed)

Math operations revert in cases of over- and underflows in Solidity starting from version 0.8.0. We recommend removing the **SafeMath** library from the **vSwapERC20** contract to optimize gas consumption and simplify the logic of the project.

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

L02. Dependency management issues (fixed)

We have spotted several issues regarding the project setup:

- Optimization is not turned on in **truffle-config.js**. This may lead to an unpredictable compilation of the project.
- **package-lock.json** is in **.gitignore**. This file is of great importance as it helps to set up the project correctly for the audit process.
- Project dependencies are in the `dependencies` rather than the `devDependencies` section of the **package.json**.
- OpenZeppelin libraries **Math**, **SafeERC20**, and **SafeMath** are copied into the project repository but should be managed as dependencies.

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

L03. Code quality - constants (fixed)

Consider declaring literals as `constants` where possible in order to increase code readability.

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

L04. Code quality - value (fixed)

Consider assigning variables to `1e18` instead of `ether` when working with tokens in order to increase code readability.

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

L05. Code quality - calculations (fixed)

Performing division operations before multiplication can result in a loss of precision. Consider dividing in the last step in these functions of the **vSwapMath** contract:

- `calculateLPTokensAmount;`
- `calculateReserveRatio;`

And in the `burn` function of the **vPair** contract.

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

L06. Code quality - import (fixed)

Consider removing one of the imports of the `IvPairFactory` file as it is imported twice in the **vRouter** contract.

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

L07. Code quality - redundant function (fixed)

Consider removing the redundant `constructor` function of the **vSwapERC20** contract as it does not contain any logic inside.

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

L08. Code quality - redundant contract (fixed)

Consider removing currently unused contracts from the project in order to increase code readability.

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

L09. Code quality - redundant check (fixed)

The check with `if` statement in `transferFrom` function of the **vSwapERC20** contract is redundant as the operation with zero allowance will revert in the next substitution.

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

L10. Code quality - external (fixed)

Consider declaring functions as `external` instead of `public` when possible to improve code readability.

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

L11. Code quality - dead code (fixed)

There are a lot of dead code and redundant commented functions all over the code. Consider removing them in order to increase code readability.

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

Notes

N01. Design drawback

Consider using `create2` instead of `create`. By using `create`, the project lacks the ability to pre-calculate the pair address by the set of tokens. Instead, it calls the factory contract to get the pair address. By using `create2`, the project gains this ability. Avoiding an external call is crucial in terms of gas-saving.

Comment from the developers: Calculating deterministic addresses with `create2` is planned for the next release.

This analysis was performed by Pessimistic:

Evgeny Marchenko, Senior Security Engineer

Daria Korepanova, Security Engineer

Nikita Kirillov, Junior Security Engineer

Irina Vikhareva, Project Manager

Alexander Seleznev, Founder

July 25, 2022