

PolyZap

smart contracts audit report

Prepared for:
polyzap.finance

Authors: HashEx audit team
April 2021

Contents

Disclaimer	3
Introduction	4
Contracts overview	4
Found issues	5
High severity issues	5
Medium severity issues	5
Low severity issues and general recommendations	5
Conclusion	6
References	6
Appendix A. Issues' severity classification	7
Appendix B. Types of known issues	7

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Introduction

HashEx was commissioned by the PolyZap team to perform an audit of PolyZap smart contracts. The audit was conducted between April 28 and April 29, 2021.

The audited code is located in PolyZap github repository. The audit was performed after the commit [05fb036](#). The same code is deployed to Polygon network (previously Matic) at the addresses of [0x4aAEC1FA8247F85Dc3Df20F4e03FEAFdCB087Ae9](#), [0x34De5ce6c9a395dB5710119419A7a29baa435C88](#).

The very limited documentation is available on [docs.polyzap.finance](#).

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts.
- Formally check the logic behind given smart contracts.

Information in this report should be used to understand the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

We found out that the code is a fork of UniswapV2. An audit for Uniswap is available [\[1\]](#). For this reason, we focused on the unaudited parts of code, as well as modifications made by PolyZap.

Contracts overview

`PolyZapERC20.sol`

Implementation of ERC20 token standard with additional permission functionality.

Allows the unlimited token allowance if set to `uint(-1)`.

The same as UniswapERC20.

`PolyZapFactory.sol`

The same as UniswapFactory.

`PolyZapPair.sol`

The same as UniswapPair.

`PolyZapRouter.sol`

Same as UniswapV2Router02.

Libraries & Interfaces

The same as Uniswap's ones.

Found issues

High severity issues

No high severity issues were introduced with changes in the forked code.

Medium severity issues

No medium severity issues were introduced with changes in the forked code.

ERC20 implementation

[PolyZapERC20.sol](#) lacks `increaseApproval()` and `decreaseApproval()` functions. These functions mitigate frontrun attacks on the `approve` function if a user wants to alter previously approved amounts in one transaction (see [2]). This issue is inherited from the source of the fork.

Low severity issues and general recommendations

No low severity issues were introduced with changes in the forked code.

Conclusion

Audited contracts are a copy of the UniswapV2. No high severity issues were found. One medium severity issue was found regarding the known frontrun approve function attack on ERC20 tokens.

Audit includes recommendations on code improvement.

References

1. [Uniswap audit](#)
2. [Attack vector on approve method](#)

Appendix A. Issues' severity classification

We consider an issue critical if it may cause unlimited losses or breaks the workflow of the contract and could be easily triggered.

High severity issues may lead to limited losses or break interaction with users or other contracts under very specific conditions.

Medium severity issues do not cause the full loss of functionality but break the contract logic.

Low severity issues are typically nonoptimal code, unused variables, errors in messages. Usually, these issues do not need immediate reactions.

Appendix B. List of examined issue types in the changes made to the forked code

Title	Check passed
Business logic overview	✓
Functionality checks	✓
Following best practices	✓
Access control and authorization	✓
Reentrancy attacks	✓
Front-run attacks	✓
DoS with (unexpected) revert	✓
DoS with block gas limit	✓
Transaction-ordering dependence	✓
ERC/BEP and other standards violation	✓
Unchecked math	✓
Implicit visibility levels	✓
Excessive gas usage	✓
Timestamp dependence	✓
Forcibly sending ether to a contract	✓
Weak sources of randomness	✓
Shadowing state variables	✓
Usage of deprecated code	✓