SAFETIN AUDIT

OLYMPUS MIGRATION CONTRACT

February 18th, 2022

TABLE OF CONTENTS

SUMMARY

This report was written for Olympus' migration contract in order to find flaws and vulnerabilities in the Olympus' migration contract project's source code, as well as any contract dependencies that weren't part of an officially recognized library.

The audit is based on the code of the following BSC testnet smart-contract : TokenMigrator.sol (https://pastebin.com/dfjfrRb9)

A comprehensive examination has been performed, utilizing Static Analysis, Manual Review, and Olympus' migration contract Deployment 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
- Through line-by-line manual review of the entire codebase by industry expert

OVERVIEW

UNDERSTANDING

The Olympus' Migration Contract is used to trade tokens from a fixed withdrawal date called startTime. It features a 5% bonus if the tokens are deposited before a certain date, called bonusTime. bonusTime can be changed, in order to postpone the date from which the bonus will end.

FINDINGS

1 | Use ‘immutable’ instead of ‘constant’

Severity: Informational

The startTime variable is set as immutable. Immutable means that this variable is a constant that can be instantiated in the constructor. Here the variable is hard-coded, which means that startTime could be set as constant without changing the contract’s logic.

Using the immutable keyword instead of constant will result in a lack of optimisation here as it is more computationally expensive - we therefore recommend changing this keyword to constant. For more information, see https://docs.soliditylang.org/en/v0.6.5/contracts.html#immutable .

2 | Use uint256 for decimals variable

Severity: Informational

Following industry standards, decimals variable should be uint8 and not uint256.
For more information, see :
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol.

3 | Typo in error message

Severity: Informational

The error messsage of line 48 "Migrator::must be after old bonustime" contains the following typo 'bonustime' should contain a space. The error message should be "Migrator::must be after old bonus time". We recommend using the clearest possible error messages for project development/maintenance and the user interface.

4 | Inconsistent use of allowance mechanism

Severity: Minor

The depositAll function includes a check via the standard allowance mechanism, but not the deposit function. We recommend removing this check from depositAll as it consumes unnecessary gas (If the amount transferred were too large the transfer would fail and the function would stop, so there is no need to add a conditional structure).

5 | Unlocked compiler version

Severity: Minor

Olympus' migration contract does not have a locked compiler version, meaning a range of compiler versions can be used. This can lead to differing bytecodes being produced depending on the compiler version, which can create confusion when debugging as bugs may be specific to a specific compiler version(s).

To rectify this, we recommend setting the compiler to a single version, the lowest version tested to be compatible with the code, an example of this change can be seen below.

Before	After
pragma solidity >= 0.8.0<0.9.0;	pragma solidity 0.8.0;

CONCLUSION

No major issue has been found in the Olympus' migration smart-contract. The findings we reported are low severity issues. The overall security of the smart- contract is very good, the only point that should be improved is the contract code's abidance to best practices.

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 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 without Safetin's prior written consent.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 Safetin 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

Safetin security assessment 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. Safetin's position is that each company and individual are responsible for their own due diligence and continuous security. Safetin'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 fun.