Minto

MYWISH

SMART CONTRACTS EXPERTISE — RIGHT WAY TO SUCCESS
MINTO audit report

If you have any questions concerning smart contract design and audit, feel free to contact audit@mywish.io

Content

1. Overview

MINTO team asked us to perform a review of their contract. We performed a review of their code from and published this document as a write-up of our findings.

1.1 Terms of Reference for the creation of a smart contract

1.1.1 Contract details:

- Minto staking contract with reinvestment.
- Users who own BTCMT token can make deposits to the contract and receive rewards also in BTCMT token. The reward is paid by certain addresses (back) daily (no more than once a day) and distributed by contract between users in accordance with their shares from the total pool of deposits.
- For each user, the HBTC reward is immediately reinvested, i.e. added to his already existing stake
- Also, the contract has a disabled / enabled bonus system, which works as follows:
- let variable bonusStatus == (true, [10,15,20,30,40], [10, 20 , 30, 50]);
- This means that the bonus system is active (true) and as soon as the user holds his stake for 10 days or more (bonusStatus.day[0] == 10), he will start receiving bonuses (first 1%, then 2, etc.; PS: DENOMINTOR == 1000)
- To receive a bonus means, when distributing a reward, to receive an additional percentage of the due reward. So, if a user has staked for 10 days and on the 10th day he is entitled to a reward of n BTCMT, he will receive n + n/100.

1.1.2 Contract functions:

rewardTokenDonation	allows an authorized address from the backend to make rewards, which are converted from HBTC into BTCMT through the exchange and then written into the contract as a reward.
stake	allows the users to stake tokens on the staking contract or increase an existing stake.
unstake	allows users to withdraw all or partly of their stake from the contract in BTCMT tokens.
recalculate	allows you to recalculate or reinvest the user's reward in the contract.

setBonusVars	allows the owner of the contract to change the status and parameters of the bonus program.
addBonusTokens	allows the owner of the contract to deposit bonus tokens to the balance of the contract.
withdrawBonusTokens	allows the contract owner to withdraw bonus tokens from the contract.
updateHistory	allows anyone to update the history of bonuses in a contract.
setSlippage	allows the owner of the contract to change the slippagePercent from 10 to 1000 points.
addOrRemoveBack	allows the owner of the contract to change the status of the authorized address from the backend (add or remove it from the list of addresses that can make a reward).

2. Introduction

2.1. Authenticity

The contracts audited are a subset of the contracts compiled and deployed in Heco blockchain.

2.2. Scope

The audit reviewed contract source code from hecoinfo.com Contracts were reviewed in the context of the flattened file, which included solidity files. The review performed did not assess any scripts, tests, or other non-Solidity files.

2.3. Methodology

This audit was performed as a comprehensive review of the codebase and takes into consideration both the Solidity code, as well as the target platform: Heco network. The Solidity was reviewed not just for common vulnerabilities and antipatterns, but also for its parity with the intent of the deployer, for its efficiency, and for the practices used during development.

2.4. Description of the complex of procedures for reviewing the smart contract

2.4.1 Primary architecture review

- Checking the architecture of the contract.
- The correctness of the code.
- Check for linearity, shortness, and self-documentation.
- Static verification and code analysis for validity and the presence of syntactic errors.

2.4.2 Comparison of requirements and implementation

- Checking the code of the smart contract for compliance with the requirements of the customer code logic, writing algorithms, matching the initial constant values.
- Identification of potential vulnerabilities

2.5. Risk Assessment

Findings were categorized using a risk rating model based on the OWASP method. Each vulnerability takes into consideration the impact and likelihood of exploitation, as well as the relative ease with which the vulnerability is resolved; findings that permeate throughout the codebase will require much more review and work to solve and are rated higher as a result.

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology:

• Likelihood represents how likely a particular vulnerability is to be uncovered and exploited in the wild;

• Impact measures the technical loss and business damage of a successful attack;

• Severity demonstrates the overall criticality of the risk;

Likelihood and impact are categorized into three ratings: H, M and L, i.e., high, medium and low respectively. Severity is determined by likelihood and impact and can be classified into four categories accordingly, i.e., Critical, High, Medium, Low shown in following Table

Table: Vulnerability Severity Classification

impact				
	High	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**
		High	*Medium*	*Low*

2.6. Disclaimer

This document reflects the understanding of security flaws and vulnerabilities as they are known to MyWish, and as they relate to the reviewed project. This document makes no statements on the viability of the project or the safety of its code. This audit does not represent investment advice and should not be interpreted as such.

Likelihood

3. Findings

3.1. Critical Severity

No critical-severity vulnerabilities were found.

3.2. High Severity

No high-severity vulnerabilities were found.

3.3. Medium Severity

3.3.1. No medium-severity vulnerabilities were found

3.4. Low Severity

3.4.1 Function: rewardTokenDonation()

Lines: - 121

Issue description: Approve to the router happens every day

Recommendations: Check if there is enough allowance and increase if necessary or create external function that would approve tokens to the router.

3.4.2 Function: stake()

Lines: - 163-165, 203-211, 217

Issue description: Using self realization of set.

Recommendations: Consider using OpenZeppelin EnumrableSet for address type.

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/structs/EnumerableSet.sol#L214

With this you can use convenient and save .length() .add() .remove() methods

3.4.3 Function: updateHistory()

Lines: - 327

Issue description: Lack of event.

Recommendations: Please consider adding an event after changing global bonusStatus.

3.4.4 Function: setBonusVars()

Lines: - 250

Issue description: Lack of event.

Recommendations: Please consider adding an event after changing global bonusStatus.

3.4.5 Function: updateHistory()

Lines: - 321

Issue description: Lack of event.

Recommendations: Please consider adding an event after the for loop with indication of updated time period and new values.

3.4.6 Function: addOrRemoveBack()

Lines: - 314

Issue description: Lack of event.

Recommendations: Please consider adding an event about addition or removal of can-deposit status.

3.4.7 Function: setSlippage()

Lines: - 305

Issue description: Lack of event.

Recommendations: Please consider adding an event containing new slippage value

4. Documents and Resources

4.1. References

Deployed contract: https://hecoinfo.com/address/0xe742FCE58484FF7be7835D95E350c23CE55A7E12

OWASP. Risk Rating Methodology. https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

5. Conclusion

The information in this review is a list of recommendations on what needs to be done to ensure the quality and security of the smart contract. The MyWish experts conducted the verification of the smart contract. Based on the results of the reviewing and testing, it is established that the token smart contract complies with the specifications specified in the terms of reference.

During the reviewing and testing of the contracts, critical errors and possible vulnerabilities were not detected so the contract is ready to be used in HECO and BSC blockchains. Outside of the included notes, the code reviewed was simple and clean. The formatting, naming, and other conventions used were fairly regular, and the inheritance structure was well-organized, resulting in a codebase that was easier to review. audit@mywish.io